VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro generování rostlin a stromů

Tree and Plants Modeling Software

2014 Daniel Molko

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Daniel Molko**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Nástroj pro generování rostlin a stromů
Tree and Plants Modeling Software

Zásady pro vypracování:

V současné době existuje velká řada modelovacích nástrojů pro tvorbu 3D scén a to jak komerčních tak nekomerčních. Součástí této práce je vytvoření nástroje, který bude umožňovat na základě existující scény generovat rostliny, keře a stromy. Generované objekty budou závislé na nastavení scény a parametrů uživatele (gravitace, výška apod.).

1. Nastudujte problematiku fraktálu a generování rostlin a stromů.
2. Vytvořte nástroj umožňující načíst existující scénu a umožňující vygenerovat rostliny, keře a stromy, s možnosti jejich výsledné editace.
3. Vyberte vhodný formát, ve kterém půjde výsledná scéna uložit tak, aby ji bylo možné importovat do dalších modelovacích nástrojů.
4. Ve scéně se zaměřte na možnosti kolizí jednotlivých objektů.
5. V případě generovaných objektů půjde navolit také vlastnosti jednotlivým částem, jako jsou například jednotlivé textury, koncové listy stromů, květy rostlin, plody stromů apod.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Němec, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

Rád by som na tomto mieste poďakoval všetkým, ktorí mi s prácou pomohli, pretože bez nich by táto práca nevznikla. Predovšetkým by som chcel poďakovať svojmu vedúcemu diplomovej práce.

Abstrakt

Táto diplomová práca sa zaoberá tvorbou 3D modelov stromov a rastlín. V teoretickej časti je uvedený historický vývoj problematiky týkajúcej sa tvorby 3D modelov stromov. Taktiež obsahuje základný popis a ukážku fraktálov a fraktálových štruktúr. Následne sú popísané rôzne metódy generovania 3D modelov stromov. Nasleduje kapitola, kde je popísaná implementácia programu pre tvorbu 3D modelov stromov. Ďalej sú uvedené výsledky vykonaných meraní a porovnaní dvoch implementovaných metód a ukážky vygenerovaných modelov oboma implementovanými metódami. V závere práce sú zhrnuté získané výsledky a predložený návrh budúceho vývoja programu.

Klíčová slova: modely vegetácie, 3D modely stromov, generovanie stromov, 3D grafika

Abstract

This diploma thesis focuses on creating 3D models of trees and plants. The theoretical part describes historical development of 3D trees modelling issues. Also contains basic description and illustration of fractals and fractal structures. In consequence the thesis describes several techniques of generating 3D models of trees. Next chapter describes an implementation of program creating 3D models of trees. The following chapter brings results of measurements and comparisons of two implemented techniques and examples of models created by both implemented methods. Resume brings a summary of obtained results and suggestion for software development.

Keywords: models of vegetations, 3D models of trees, generating trees, 3D graphics

Seznam použitých zkratek a symbolů

CD	–	Compact Disc
AP	–	Attraction Point
BP	–	Base Point
L-systém	–	Lindenmayer systém
3D	–	3 Dimenzion

Obsah

1 Úvod

Počítačovo vytvorená grafika, sa dnes používa skoro vo všetkých zobrazovacích prostriedkoch. Či už ide o film, hru, plagát alebo obrázok v časopise, všade sa čoraz častejšie využívajú počítačovo generované modely. Takýto postup má svoj veľký ekonomický význam. Ak chcem napríklad v časopise použiť fotku palmy na púšti, nemusím ísť na púšť a odfotit si ju, ale stačí sadnúť za počítač a vymodelovať danú scénu v počítači. Moderné filmy (hlavne scifi) sa už nenátačajú v štúdiach s prepracovanými kulisami z kartónu, ale väčšinou pred zeleným plátnom, miesto ktorého sa následne v počítači vygeneruje celá scéna. Počítačové hry sú osobitnou kapitolou, pretože bez prepracovanej a vierohodnej grafiky by len ťažko obstáli v modernej silnej konkurencii. Moderné grafické karty a výkonné viacjadrové počítače, nám dnes umožnujú generovať veľmi realisticky vyzerajúce modely vecí v reálnom čase. Preto toto odvetvie informačných technológii napreduje míľovými krokmi.

V mojej diplomovej práci sa zaoberám tvorbou virtuálnych modelov stromov a rastlín. V úvode sa rozoberám problematikou tvorby virtuálných modelov vegetácie od jej začiatkov až k moderným metódam. Rozoberám taktiež problematiku fraktálov, ktoré boli často použité ako hlavná metóda generovania virtuálnych stromov a rastlín. Niektoré generovacie programy ich v zložitejšej forme používajú dodnes. V ďalšej časti sa zaoberám modernými metódami tvorby realisticky vyzerajúcich modelov stromov a rastlín. Ide o metódy simulujúce biologický rast stromov využívajúcich metódu konkurenčného boja o priestor. Ďalej je to metóda získavania 3D modelu stromu z obrázku a na záver ver je to fraktálna metóda (L-systém), ktorá bola modifikovaná a boli v nej použité prvky evolučných algoritmov.

Cielom mojej diplomovej práce je vytvorenie programu, ktorý bude generovať realistické 3D modely stromov a kríkov. Užívateľ bude môcť zadať vstupné parameter ako tvar koruny, hrúbka konárov či rozvetvenosť stromu a na základe nich, program vygeneruje požadovaný strom, prípadne krík. Ak bude zvolená daná možnosť, model bude vygenerovaný aj s listami, ktorých počet a veľkosť bude môcť užívateľ zmeniť podľa svojej vôle. Výsledný model stromu s listami bude otextúrovaný predvolenou textúrou, avšak túto textúru si bude môcť užívateľ zmeniť a tým prispôsobiť model určitému druhu stromu. Výsledný model bude možné vyexportovať do súboru a následne naimportovať do niektorého grafického programu (blender, 3D studio max), kde sa bude môcť dodatočne upravovať alebo použiť v požadovanej scéne. V poslednej kapitole budú vykonané experimenty a budem v nich poorovnávať dve popísané metódy rastu stromov. Najskôr vykonám merania na oboch metódach osobitne a následne budú vykonané merania s porovnávaním výsledkov pre obe metódy. Experiment má odhaliť silné a slabé stránky každej metódy a tým určiť vhodnosť danej metódy pre danú situáciu. V závere práce budú zhrnuté všetky dosiahnuté výsledky.

2 Modelovanie stromov a rastlín

Modelovanie stromov a rastlín pomocou počítača má pomerne dlhú históriu. Prvý model bol navrhnutý Stanislawom Ulamom a využíval koncept celulárneho automatu, ktorý bol predtým navrhnutý von Neumanom a Ulamom. Pricíp vetvenia bol vytvorený iteratívne, počnúc jednofarebnou bunkou v trojuholníkovej mriežke. Následne sa zafrabili ďalšie bunky, ktoré sa dotýkali jedného vrcholu bunky, zafarbenej v predchádzajúcom kroku iterácie.

Táto základná myšlienka podnietila vznik ďalších rozšírení. Meinhardt nahradil trojuholníkovú sieť štvorcovou a výsledný bunkový priestor používal na skúmanie biologických hypotéz ohľadom tvorby sieťovej štruktúry. Jeho výsledky pomohli objasniť vývoj štruktúry žíl v liste rastlín.

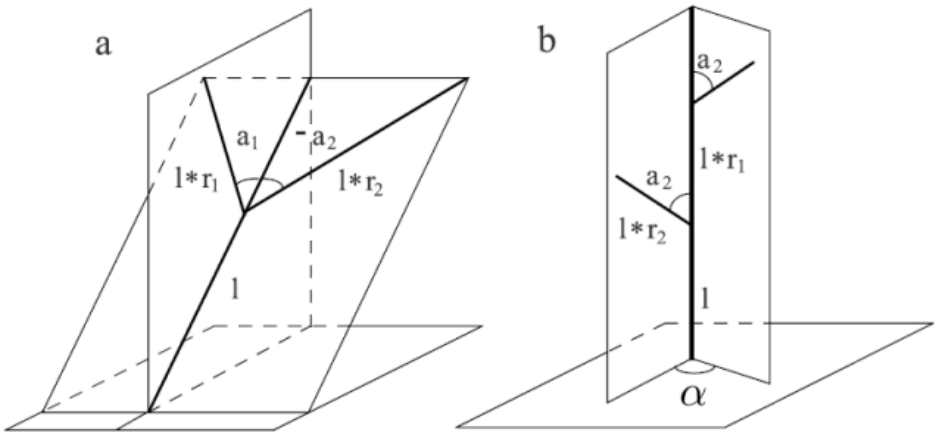
Greene rozšíril celulárny automat do troch rozmerov a výsledky z neho použil na simulovanie rastových procesov rastlín a stromov, reagujúce na okolité prostredie. Simuloval tak napríklad rast viniča po povrchu domu.

Cohen simuloval vývoj vetvenia pomocou rozširujúcich pravidiel, ktoré sa sústavne aplikujú na celý systém. Jeho systém bol omnoho flexibilnejší a dynamickejší než diskrétny systém bunkového 3D priestoru navrhnutého predtým.

Spoločným rysom týchto postupov je dôraz na interakcie medzi rôznymi prvkami rastúcej štruktúry, rovnako ako aj dôraz na samotnú štruktúru a prostredie. Hoci interakcie zobrazujú veľmi reálne rast rastlín, náročnosť modelu je zložitejšia. Preto jednoduchšie modely ignorujú základné faktory ako sú kolízie s prostredím alebo kolízie medzi samotnými výrastkami navzájom. Prvý model v tejto kategórii navrhol Kotaro Honda. Slúžil mu na študovanie stromov, za použitia nasledovných pravidiel:

- Segmenty stromu sú rovné a ich obvod sa nebere do úvahy.
- Rodičovský segment produkuje dva dcérske segmenty prostredníctvom jedného vetvenia.
- Dlýžky dvoch dcérskych segmentov sú skrátené o konštantné hodnoty r_1 a r_2 s ohľadom na rodičovský segment.
- Rodičovský segmet a jeho dva dcérske segmenty sú v rovnakej rovine vetvenia. Dcérske segmenty sú pootočené o konštantné hodnoty uhlov a_1 a a_2 s ohľadom na rodičovský segment.
- Rovina vetvenia je stanovená vzhľadom k smeru gravitácie tak, aby bola čo najbližšie k vodorovnej rovine.
  Výnimkou sú výrastky pripojené k hlavnému kmeňu. V tomto prípade je konštantný divergentný uhol α nemenný.

Zmenami číselných parametrov získal Honda širokú škálu rôznych tvarov stromov. Po určitých vylepšeniach bol jeho model aplikovaný na skúmanie vývoja rozvetvovacích vzorov skutočných stromov. Následne boli navrhnuté rôzne ďalšie pravidlá pre vetviace uhly, v ktorých roviny postupných vetvení sú na seba navzájom kolmé. Hondov autom



Obrázek 1: Ukážka stromovej geometrie, ktorú vyvinul Honda

slúžil ako základ pre stromové modely navrhnuté Aonom a Kunim. Navrhli niekoľko vylepšení Hondovho modelu, z ktorých najvýznamnejšie bolo uprednostnenie vedľajších vetvení v určitom smere, použité na vytvorenie účinkov vetra, fototropizmu (schopnosť mnohých rastlín i niektorých živočíchov obracať sa smerom k zdroju svetla) a gravitácie.

Modely Hondy, Aonoa a Kuniho boli repreznetované pomocou priamiek s konštantnou alebo rôznou šírkou. Používali sa hlavne na tvorbu kostry stromov. Podstatné zlepšenie realizmu modelov stromov, priniesli Bloomenthal a Oppenheimer. Zaviedli zakrivenie vetiev, starostlivo modelované plochy okolo rozvetvení a textúry kôry a listov.

Nové návrhy vychádzali z Hondovho automatu a zaviedli sa stochastické (náhodné) štruktúry definované pomocou deterministických algoritmov. Hlavnými predstavitelmi, ktorí vyvíjali takéto modeli boli Reeves a Blau, de Reffye a Remphrey, Neal a Steeves. Hoci tieto modely boli popísané pomocou rôznej terminológie, zdieľajú základné paradigma upresňujúce stromovú štruktúr, ohľadom pravdepodobnosti, s ktorou sú tvorené vetvy. Niektorí sa zameriavali na modelovania stromu ako tvaru, bez bilogocikých podrobností. Iní sa venovali modelovaniu podrobných detailov a vytvorenie čo najrealisteckejších stromov, čo dosahovali modelovaním vývoja púčika v určitých časových intervaloch.

Vývoj bol ralizovaný nasledovnými možnými stavmi:

- Nerob nič.
- Zmeň sa na kvet.
- Zmeň sa na jedne alebo viac listov vyrastajúcich z umiestnenia púčika.
- Zomri a zmizni.

Tieto stochastické preddpisy sa stali základnými pravidlami pre modelovanie 23 druhov štruktúr stromov. Geometrické parametre ako sú dĺžka a priemer vetvy, uhol vetvenia a podobne sú taktiež stochastické.

L - systém poprvý krát aplikovali na generovanie stromov Aono a Kuni. Použili originálnu definíciu L - systému a dospeli k záveru, že je nevhodná pre modelovanie kompexných vetviacich vzorov vyšších rastlín. Avšak ich argumenty sa nevzťahovali na parametrický L - systém s koritnačovou realizáciou.

3 Fraktály

Fraktál je rekurzívne vybudovaný geometrický objekt. Ide o nepravidelný, fragmentovaný (rozdelený) geometrický tvar, ktorý po rozdelení na menšie časti, vykazuje tvarovú podobnosť s týmito časťami. Disciplína zaoberajúca sa fraktálmi sa nazýva Fraktálna geometria a za jej zakladateľa sa považuje Benoít Mandelbrot. Aj pred zavedením Fraktálnej geometrie sa vedci a umelci zaoberali geometrickými útvarmi, ktoré dnes nazývame fraktály, ako napríklad Kochova snehová vločka alebo Mandelbortova krivka.

Základné rozdelenie fraktálov:

Dynamické systémy s fraktálovou štruktúrou: Stav dynamického systému v čase je reprezentovaný stavovým vektorom, ktorého zmena zložiek je vyjadrená sústavou diferenciálnych rovníc. Takýto dynamický systém môže vykazovať vlastnosti deterministického chaosu. Zobrazením stavového vektora vzniká fraktálový obrazec pomocou ktorého môžeme skúmať chaotické vlastnosti daného dynamického systému. Príkladom pre takýto systém je model dynamiky rastu obyvateľstva.

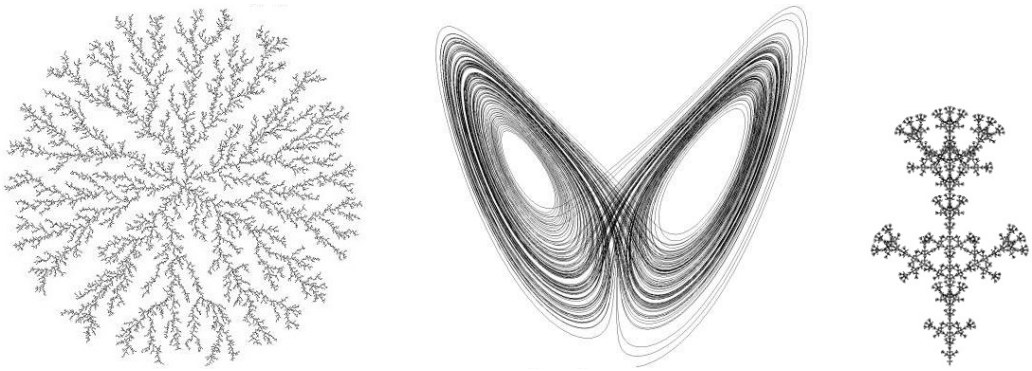
L-systémy: Teória Lsystémov vznikla zo známeho jazyka pre výuku programovania LOGO, kde korytnačka kreslí na obrazovku rôzne obrazce. Za Lsystém sa dá označiť určitá formálna gramatika pre generovanie nejakého obrazu pomocou terminálnych symbolov forward, back, left, right, push a pop. Využitie Lsystémov je v oblasti generovania obrazov umelých rastlín (teda je možne vykresliť realistický obrázok stromu iba na základe takejto gramatiky)

Systémy iterovaných funkcií IFS: V tejto metóde sa fraktál vykresluje momocou tzv. generatívnej metódy. Fraktál môže byť plne deterministický alebo funkcia, ktorá ho generuje môže obsahovať aj náhodné prvky. Pomocou tejto metódy môžeme zobrazovať napr. dym.

Stochastické fraktály (nepravidelné fraktály): Na generovanie tohto typu fraktálu sa do značnej časti podieľa náhoda. Jedným z možných spôsbov generovania je simulácia Brownovho pohybu, metóda presúvania stredného bodu alebo spektrálna syntéza.

V počítačovej grafike sú fraktálne štruktúry čoraz častejšie využívané. Ide o pomerne jednoduchý ale pritom veľmi efektívny druh tvorby grafiky. Dajú sa implementovať ako v dvojrozmernom tak aj vo viacrozmerných priestoroch. Na tvorbu fraktálov sa v počítačovej grafike nahviac využívajú: L-systém (Lindenmajerov systém), IFS (Iterated Function system) alebo Stochastické fraktály. V našom prípade, pri generovaní stromov a rastlín, sa najviac hodia L-systém a Stochastické fraktály a preto sa na tieto dva systémy zameriamy a budeme ich navzájom porovnávať.

Ukážka fraktálových útvarov



Obrázek 2: Ukážky fraktálnych útvarov s využitím rôznych technik

4 Možnosti generovania

4.1 L-systémy

Definícia 1

L - systém *je definovaný ako trojica* $G = (\Sigma, R, A)$*, kde* Σ *je abeceda, t.j. konečná neprázdna množina znakov, R je množina prepisovacích pravidiel a* $A \in \Sigma*$ *je axióma – konečné slovo nad abecedou* Σ*, v ktorom odvodenie začína. [6]*

Otcom L – systému bol Maďarský biológ Aristid Lindenmayer, podľa ktorého dostal aj pomenovanie: Lindenmayerov (skrátene L) systém. Roku 1968 navrhol model, pomocou ktorého bolo možné simulovať rast jednoduchých vláknitých biologických štruktúr ako sú reťazce buniek, neuróny a podobne. Jeho model vychádza z modelu formálnych gramatík a obsahuje abecedu, pravidlá a axióm.

Za základný útvar L – systému sa považuje Kochova vločka. Na začiatku bola úsečka, ktorú rozdelil na 3 časti. Nad prostrednou časťou zostrojil rovnostranný trojuholník, ktorému vymazal preponu. Na každej rovnej úsečke opakoval rovnaký postup. Výsledkom bola teoreticky nekonečne dlhá krivka s tvarom pripomínajúcim polovicu snehovej vločky. Zápis pre zostrojenie Kochovej vložky môže vypadať nasledovne:

X –> X + FX – FX + FX

Axiom = FX

kde X symbolizuje pravidlo, F rast dopredu, + a – otočenie smeru rastu.

Teraz môžme sledovať, ako sa bude axiom postupne v krokoch vyvíjať:

4.1.1 Gramatika a pravidlá

Ako sme už vyššie spomínali, L – systém sa skladá z abecedy, pravidiel a axiomu a vychádza z formálnych gramatík. Abeceda je konečná množina symbolov, ktorá tvorí gramatiku jazyka. Všetky ostatné symboly nepatriace do tejto množiny, budú našim systémom ignorované. Pre náš konkrétny projekt som zvolil nasledovnú množinu: (X, Y, Z, F, x, y, z, f, +, -, [,]).

Popis významu jednotlivých symbolov:

X, x, Y, y, Z, z : pravidlá systému.

F, f : indikuje rast štruktúry o jednu jednotku.

+ : indikuje otočenie vektora rastu o hodnotu uhla v smere pohybu hodinových ručičiek.

- : indikuje otočenie vektora rastu o hodnotu uhla proti smeru pohybu hodinových ručičiek.

] : uloží do pamäte pozíciu bodu, v ktorom sa práve nachádza a začnú sa vykonávať nasledovné symboly bez ohľadu na predošlé nastavenia. Vytvorí sa akoby bod, od ktorého sa tvorí nová podštruktúra systému.

[: vráti sa k bodu uloženému pri vykonaní symbolu [a načíta pôvodné nastavenia (uhol, starnutie, ...).

Pravidlá sa skladajú z ľavej a pravej strany. V našom prípade, sa na ľavej strane môžu nachádzať iba symboly (X, x, Y, y, Z, z). Pravidlá určujú, ako sa majú prepisovať sym-

boly na ľavej strane pravidla. Spôsob prepisu je na pravej strane. Návrhom pravidiel určujeme tvar, orientáciu a veľkosť celej štruktúry. Pri návrhu používame iba abecedu jazyka. Axiom je počiatočný bod systému. Je to základné slovo, z ktorého sa vyvíja celý systém. Axiom nesmie byť prázdne slovo, pretože by sa systém nezačal rozvíjať. Obvykle sa skladá z kombinácie pravidiel, dodatočných symbolov a nebýva príliš rozsiahle.

Príklad použitia a rozvoja pravidiel:

(X –> X+FX [+FY]–[FZ]

Y –> ++FX –FY

Z –> –FFY-FZ

Axiom = YX)

Axiom sa bude postupne s počtom krokov rozvíjať nasledovne:

0) *Axiom* = YX

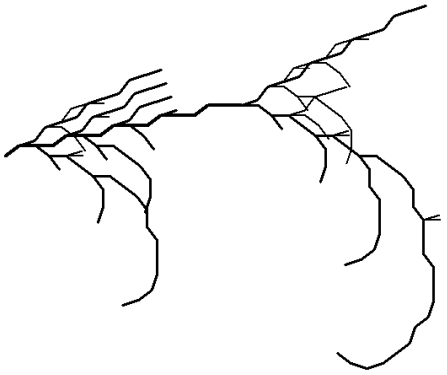
1) *Axiom* = ++FX –FY X+FX [+FY]–[FZ]

2) *Axiom* = ++F X+FX [+FY]–[FZ] –F++FX –FY X+FX [+FY]–[FZ]+F X+FX [+FY]–[FZ] [+F++FX –FY]–[F–FFY-FZ]

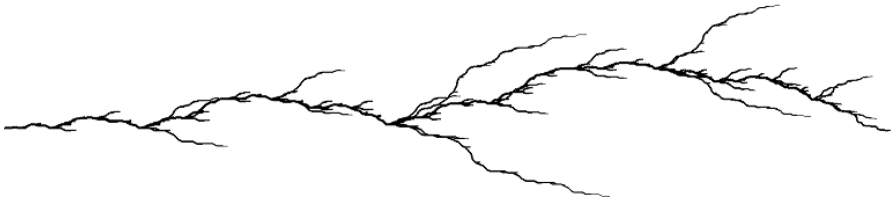
3) *Axiom* = ++F X+FX [+FY]–[FZ]+F X+FX [+FY]–[FZ] [+F++FX –FY]–[F–FFY-FZ] –F++F X+FX [+FY]–[FZ] –F++FX –FY X+FX [+FY]–[FZ]+F X+FX [+FY]–[FZ] [+F++FX –FY] ...

4.1.2 Ukážky fraktálových štruktúr

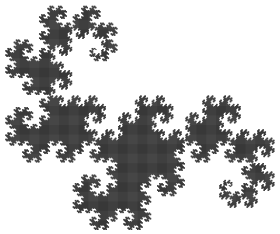
Obrázek 3: Vývoj kochovej vločky



Obrázek 4: Rozvinutý fraktál v 4. kroku



Obrázek 5:
X –> X + FX – FX + [FY] [+FZ]
Y –> FY – FY ++ FY [FY]
Z –> FZ ++ FZ – FZ [FZ]
Axiom = FX
Uhol = 12



Obrázek 6:
X –> X + YF+
Y –> -FX - Y
Axiom = FY
Uhol = 90

4.2 Samoorganozačné algoritmy

Väčšina metód pre modelovanie stromov sú založené na predpoklade, že stromy majú opakujúce sa rekurzívne štruktúry. Tento predpoklad je v súlade s ustálenou predstavou o architektonických modeloch stromov, podľa ktorej je forma stromu do značnej miery vzorom pre všetky jeho vetvenia. Typickým príkladom modelovania pomocou tohto predpokladu je použitie fraktálov. Jednu z možností (L-systém) sme popísali vyššie.

Existujú však aj iné, alternatívne metódy modelovania stromov a rastlín. Oproti fraktálom sú implementačne zložitejšie, avšak ich výsledné modely sú realistickejšie. Vytvárajú taktiež flexibilnejšie a pomocou zmeny malého počtu vstupných parametrov aj rôznorodejšie modely stromov a rastlín. Taktiež umožňujú interaktívnu manipuláciu s modelom, čo je pri zložitých fraktálnych štruktúrach výpočtovo a pamäťovo náročné.

Existujú rôzne metódy a algoritmy pre vyššie popísané modelovanie. My sa budeme zaoberať a popíšeme hlavne *space colonization algoritmus*, *self-organizing model*, *modelovanie podla predlohy z obrázka* a *modifikovaný L-systém s evolučnými algoritmami*

V nasledujúcich kapitolách si podrobne popíšeme jednotlivé metódy s ich výhodami a nevýhodami.

4.2.1 Metóda space colonization

Táto metóda modelovania stromov je zvyčajne založená na top-down systéme (môže byť však aj opačná ako si ukážeme nižšie), kedy koruna stromu rastie od listov smerom nadol ku kmeňu. Namiesto rekurzívnych algoritmov sa tu využíva predovšetkým algoritmus *konkurenčného boja o prestor*, ktorý hrá hlavnú úlohu pri určovaní tvaru stromu alebo rastliny.

Pojem *konkurenčný boj o prestor* pri modelovaní stromovej architektúry nie je žiadnou novinkou ale vychádza z predchádzajúcich rekurzívnych modelov Hondy. V roku 1962 Ulama používa tento pojem na vytvorenie 2D modelu celulárneho automatu vytvárajúceho abstraktné stromové štruktúry. Ďalšími vylepšeniami Ulamovho modelu získal Stevens vizuálne realistické modely mladých stromov.

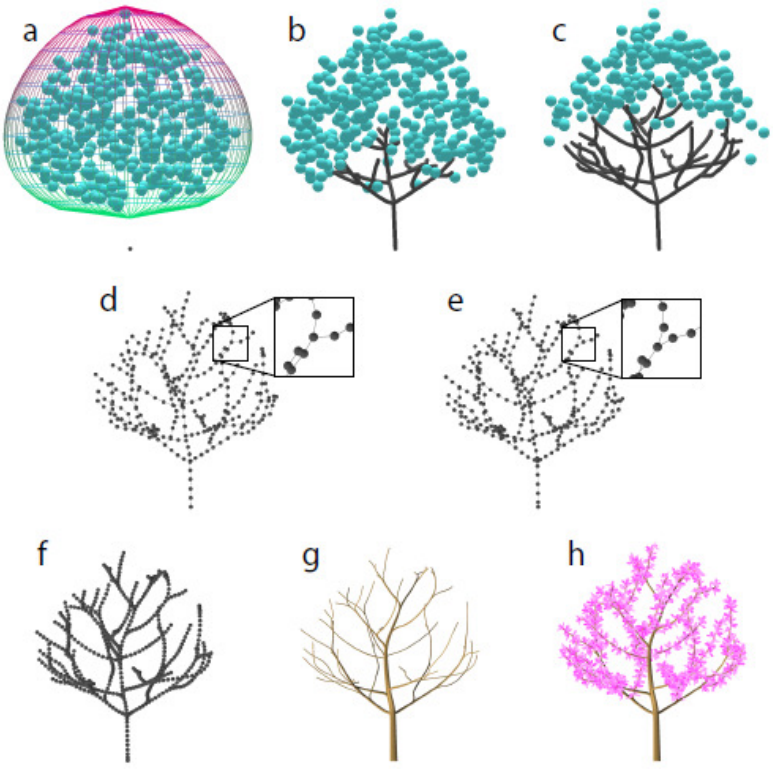
V nedávnej minulosti navrhol Rodkaew algoritmus založený na distribúcii častíc v tvare koruny stromu a následne sledovanie ich pohybu smerom dole ku koreňu. Konvergujúce cesty častíc, ktoré sú priťahované ich susedmi a základňou stromu, tvoria korunu stromu. Pomocou tohto algoritmu (a jeho rozšírení) sa dajú vytvoriť prekvapivo realistické modely stromov bez ohľadu na proces vývoja stromu.

Ďalšie rozšírenei priniesli A. Runions, B. Lane a P. Prusinkiewicz. Ich hlavnou myšlienkou bolo iteratíne pridanie nových prvkov (uzlov) do stromovej štruktúry vytvorenej v predchádzajúcich krokoch. Tento procest je riadený blízkosťou bodov, označujúcich dostupnosť voľného priestoru. Formálne vychádzali z metódy pre generovanie žíl listov navrhnutú Runionsom, ktorú rozšírili do 3D priestoru. Tvar generovaných modelov sa prispôsobeoval prítomnosti susedných rastliny a prekážok.

4.2.2 Postupy metódy

V nasledujúcich sekciach popíšeme model vytvoreny Runionsom, Laneom a Prusinkiewiczskym.

Kroky postupného vytvárania modelu sú zobrazené na obrázku 7. Trojrozmerný obal koruny stromu je vstupným parametrom algoritmu. Môže byť zadaný pomocou ľubovoľnej metódy, ktorá umožňuje ľahko testovať, či bod leží vo vnútri alebo mimo ohraničený priestor. Vhodné je použiť napríklad povrch získaný otáčaním rovinne generovanej krivky (prípadne s meniacim sa tvarom) okolo zvislej osi stromu. Na začiatku generovania stromu sa priestor vo vnútri obálky naočkuje sadou spádových bodov (obr. a). Tieto body signalizujú dostupnosť prázdneho priestoru pre rast a po spojení s vetvou sú tieto body odstránené. Rozmiestnenie spádových bodov je vstupným parametrom a určuje ho užívateľ.



Obrázek 7: Postup vytvárania modelu [2]

Následkom príťažlivosti atrakčných bodov vzniká kostra stromu iteratívnym postupom počnúc od prvého uzla na základni stromu. V tejto metóde teda rastie strom od základne k listom, teda opačne ako popisuje Rodkaew, avšak princíp ostáva rovnaký. V každej iterácii vymedzujú nové uzly krátke vetvy segmentu a tým rozširujú kostru v

smere blízkom príťažlivosti bodov (obr. b, c). Tento proces sa končí, ak všetky atrakčné body boli odstránené, ak v okruhu vplyvu zostávajúcich atrakčných bodov sa nenachádzajú žiadne uzly alebo ak bol dosiahnutý užívateľom definovaný počet iterácií.

Výsledná kostra stromu je ďalej upravovaná. Ako prvé je vhodné znížiť počet paralélnych uzlov v kostre stromu a tým sa zníži aj množstvo dát, potrebných na vytvorenie stromovej geometrie (obr. d). Ďalej sa rozdvojené uzly presunú bližšie k materskému uzlu a tým sa vyhladí uhol rozvetvenia (porovnaj obrázky c, d). Tento krok má významný dopad na celkový vzhľad kostry stromu. Rozdelenie kriviek vetviacich štruktúr aplikované na pôvodný alebo upravený model, vytvára hladšie zaoblené vetvy (obr. f). Akonáhle su tieto kroky dokončené, geometria stromu je modelovaná pomocou cylindrov umiestnených v strede osi kostry (obr. g).

Priemer vetiev je určený pomocou kombinácie priemeru pred a za rozvetveniami. Výpočet začína predpokladom, že všetky typy vetiev majú rovnaký počiatočný polomer r_0 a smer prechádzania vetiev je od koruny ku kmeňu. Ak sa stretnú dve vetvy s polomermi r_1 a r_2, výsledný polomer r novej spoločnej vetvy sa vypočíta pomocou vzorca

$$r^n = r_1^n + r_2^n \tag{1}$$

kde n je vstupný parameter metódy (zvyčajne v rozsahu hodnot 2 až 3). Ak sú potrebné orgány ako sú listy, kvety a malé vetvičky, sú pridané do stromu (obr. h) a ich rozmiestnenie sa určuje buď náhodne, alebo podľa vopred definovaných pravidiel (s ohľadom na prostredie, natáčanie listov k svetlu a podobne).

4.2.3 Algoritmus konkurenčného boja o priestor

Hlavným prvkom vyššie popísanej metódy je algoritmus konkurenčného boja o presto (alebo space colonization algoritmus), ktorého kľúčovým faktorom určujúcim vetvenie a vzhľad stromu, je boj konárov o miesto.

Algoritmus začína s počiatočnou konfiguráciou s N spádovými bodmi (obvykle stovky až tisíce) a s jedným alebo viacerými počiatočnými uzlami stromu. Strom je generovaný iteratívne. V každej iterácii môže spádový bod ovplivniť svojou príťažlivosťou uzol stromu v jeho blízkosti, ktorý prekročí polomer ovplivňovania d_i spádového bodu. Na jeden uzol stromu môže súčasne pôsobiť aj viac spádových bodov v: súbor týchto bodov budeme označovať $S(v)$. Ak $S(v)$ nie je prázdny, nový uzol stromu v' bude vytvorený a pripojený k v podľa segmentov (v, v'). Uzol v' je umiestnený do vzdialenosti D od v, v smere definovanom ako priemer normalizovaných vektorov ku všetkým zdrojom $s \in S(v)$. Platí $v' = v + D\hat{n}$, kde

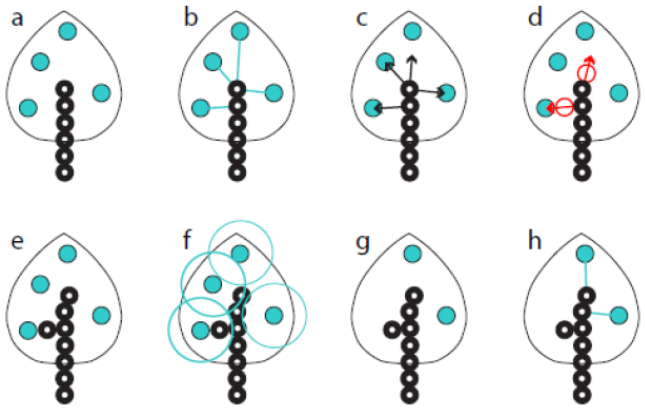
$$\hat{n} = \frac{\vec{n}}{\|\vec{n}\|} \tag{2}$$

$$\vec{n} = \sum_{s \in S(v)} \frac{s - v}{\|s - v\|} \tag{3}$$

Smer rastu môžeme ešte čiastočne ovplivniť pomocou vektora $\vec{g}$, ktorý predstavuje kombináciu hmotnosti vetvy a fototropizmu.

$$\tilde{n} = \frac{\hat{n} + \vec{g}}{\|\hat{n} + \vec{g}\|} \tag{4}$$

Akonáhle sa pridá nový uzol, vykoná sa kontrola, či má byť zmazaný spádový bod, ku ktorému smeruje rast danej vetvy. Spádový bod s bude zmazaný, ak aspoň jeden uzol v prekročil jeho mazaciu zónu d_k.



Obrázek 8: Space colonization algoritmus

Popis priebehu algoritmu (obrázok 8):

a) Začíname vo fáze, kedy sa štruktúra stromu skladá zo šiestich uzlov (čierne disky s bielymi stredy) a štyroch spádových bodov (modré disky).

b) Každý spádový bod je spojený s najbližším uzlom stromu, za predpokladu, že uzol je v okruhu jeho vplyvu (modré čiary), čo vytvára sadu spádových bodov, ktoré ovplyvňujú každý uzol.

c) Sú nájdené normalizované vektoroy z každého uzla pre každý zdroj (spádovy bod), ktorý ovplyvňuje uzol (čierne šípky).

d) Súčet všetkých týchto vektorov je znovu normalizovaný (červené šípky), poskytuje základ pre umiestnenie nových uzlov stromu (červené kruhy).

e) Nové uzly sú začlenená do stromovej štruktúry. V tomto prípade sa rozširuje hlavná os a začína sa bočná vetva.

f) Testujú sa mazacie zóny (modré kruhy) spádových bodov, či sa v nich nachádzajú stredy uzlov. V našom príklade bolo zistené, že u dvoch spádových bodov bola narušená mazacia zóna stredom novo pridaných uzlov.

g) Spádové body s narušenou mazaciou zónou su odstránené.

h) Začína sa d'alšia iterácia algoritmu prechodom na bod a).

Obrázek 9: Ukážka modelov vytvorených algoritmom *konkurenčného boja o priestor* [2]

Výhody

- Realistické modely stromov.
- Presné ovládanie tvaru stromu pomocou spádových bodov.
- Malý počet vstupných parametrov.
- Možnosť modelovania širokej škály stromov a kríkov.

Nevýhody

- Zložitejši model a jeho softvérová implementácia.
- Zložité a pri veľkých štruktúrach časovo náročné výpočty.
- Problémové stanovenie obalu koruny.

Navrhovaná metóda je užitočná najmä pri simulácii nepravidelných tvarov listnatých stromov miernej klímy. Tieto tvary sú ťažko zachytitelné staršími modelmi, ktoré zdôrazňujú rekurzívne aspekty stromovej štruktúry. Modely vytvorené pomocou algoritmu *konkurenčného boja o priestor* sú vizuálne prijateľné, aj bez listov a kvetov, ktoré by mohli maskovať nedostatky. V prípade potreby môžu byť modely ľahko doplnené o listy, kvety, púčiky, a ovocie.

4.2.4 Metóda Self-organizing

Opísaná metóda vychádza z modelu, ktorý vytvorili W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R.Měch a P. Prusinkiewicz.

Táto metóda umožňuje generovať široký rad vysoko realistických modelov stromov a kontrolovať ich tvar pomocou malého počtu parametrov a zároveň umožňuje interaktívnu manipuláciu s modelom. Nami popísaná metóda je zameraná predovšetkým na modely stromov a kríkov miernej klímy. Tieto stromy obyčajne produkujú veľa púčikov, z ktorých väčšina sa nevyvíja v trvalé konáre.

Okrem iného je potrebné stanoviť jasnú koncepciu riadenia osudu púčikov a konárov, čo vyžaduje základné vedomosti o vývoji a formovaní stromov. Základy tohto algoritmu stanovili už v roku 1995 Sachs a Novoplansky. Zdôraznili samo-organizovací charakter vývoja stromu vývoja, v ktorom "každý púčik a vetva je neustále v porovnaná s alternatívami, ktoré by mohli mať rovnakú úlohu v celkovej stromovej štruktúre."[Sachs 2004].

4.2.5 Terminológia

Uzol - Bod, v ktorom je jeden alebo viac listov pripojených ku konáru.

Medziuzlie - Časť stonky medzi dvoma *uzlami*.

Bočný púčik - Púčik, vytvorený v *uzle* (medzi listom a stonkou).

Koncový púčik - Púčik, vytvorený na konci vetvy.

Metamer - *Medziuzlie* obsahujúce list a púčik.

Púčik môže mať štyri rôzne stavy:

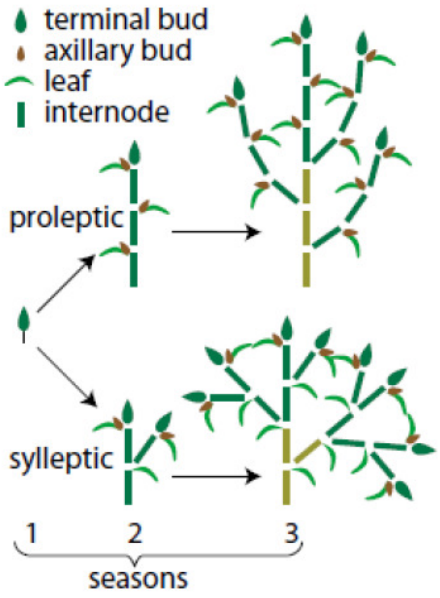
1. vytvára nový *metamer*
2. produkuje kvet
3. zostáva nevyužitý (zachováva sa možnosť rozvíjania v budúcnosti)
4. umiera

Metamery sa možu vytvárať priebežne alebo rytmicky. Rast stromov mierneho pásma je rytmický - regulovaný cyklom ročných období.

Prírastok - Postupnosť *metamerov*, vytvorených v jednom cykle (novo vyrastené výhonky na jar).

Apical - Potlačenie rastu bočných vetiev a podpora rastu koncových vetiev.

Hlavný kmeň sa vyvíja z koncového púčika umiestneného v semiačku a má index usporiadania 0. Následne vytvára bočné púčiky n na osi rastu až po koncové púčiky s indexom $n + 1$. Nové vetvy môžu byť vytvárané *syllepticky*, t.j. v rovnakom období, v ktorom sa vytvorila ich materská vetva, alebo *prolepticky*, čo znamená, že púčik bol vytvorený v roku m a vo vývoji môže pokračovať až v roku $m+1$ (obr. 10). Pre modelovanie stromov v našom klimatickom pásme je vhodnejšia metóda *proleptického* rastu.



Obrázek 10: Rast stromu proleptickym alebo sylepticky spôsobom [4]

4.2.6 Výpočet stavu púčikov

Pre výpočet stavu púčiku použijeme ako vstupný parameter životné prostredie stromu (dostupnosť voľného priestoru alebo svetla), a teda určíme, ktoré púčiky budú vyrábať nové *prírastky* a ako veľké budú. Najjednoduchší prípad je *sylleptický* vývoj, kedz sú si všetky púčiky ekvivalentné. Tento proces produkuje krátke košaté kríky. Rovnaký tvar sa dá dosiahnuť aj *proleptickým* vývojom, v ktorom môžu iba bočné púčiky vytvárať nové výhonky v budúcej sezóne, zatiaľ čo koncové púčiky vytvárajú kvetinové štruktúry alebo zomrú.

Vývoj vysokých pretiahnutých stromov, vyžaduje rast hlavne koncových púčikov a potlačenie rastu bočných púčikov. Nie je celkom jasné, či sa o štíhly rast stromov v prírode zasluhuje hormonálne riadenie, súťaž o zdroje alebo oboje. V nasledujúcej časti popíšeme dva modely využívajúce súťaž o zdroje.

Rozšírený Borchert-Honda (BH) model

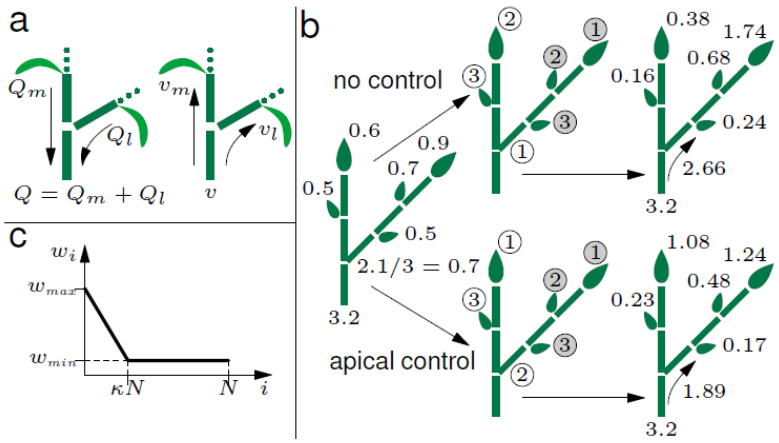
BH model bol pôvodne navrhnutý ako čisto endogénny (informácie sa šírili od kmeňa k listom) mechanizmus, ktorý reguluje rozsah vetvenia tým, že riadi distribúciu rastu rozdelovaním zdrojov medzi púčiky. Algoritmus pracuje v dvoch krokoch. V prvom kroku, zhromaždí informácie o množstve svetla Q, ktoré dosiahnú púčiky a jeho kumulatívne hodnoty sa uložia v rámci *medziuzlia* (Q_m, Q_l, Q obr. 11 a).

Súhrnná hodnota Q na základni určuje množstvo svetla, ktoré bude distribuované v druhom kroku: $v_{base} = \alpha Q_{base}$, kde α je proporcionálny koeficient. Veľkosť dosiahnutého zdroju je rozdelená medzi hlavnú os (v_m) a bočnú vetvu (v_l) (obr. 11a) pomocou rovníc:

$$v_m = v\frac{\lambda Q_m}{\lambda Q_m + (1-\lambda)Q_l} \tag{5}$$

$$v_l = v\frac{(1-\lambda)Q_l}{\lambda Q_m + (1-\lambda)Q_l} \tag{6}$$

Parameter$\lambda \in [0,1]$ určuje, či je alokácia zdrojov skreslené v prospech hlavnej osi ($\lambda > 0,5$), nie je skreslená ($\lambda = 0,5$), alebo je skreslená v prospech postrannej vetvy ($\lambda < 0,5$). Celočíselná časť množstva zdroju v priradeného púčiku určuje počet *metamerov* vyprodukovaných púčikom $n = [v]$. Dĺžky l týchto nových medziuzlov sa vypočítajú zo vzorca $l = v/n$. Parametrom λ môžme jednoducho ovplivňovať vzhlad a košatosť stromu.



Obrázek 11: Algoritmus rozšíreného BH modelu [4]

Prioritný model

Na rozdiel od BH modelu, ktorý kumuluje zdroje v kmeni rastliny a následne ich distribuuje každému púčiku, prioritný model pracuje na úrovniach modelu. Informácie o množstve zachyteného svetla púčikmy a o počte púčikov sú uložené v základni každej osi (obr. 11 b v ľavo). Priemerné množstvá dopadajúceho svetla (celkové množstvo svetla na zachyteného púčikmy na určitej vetve / množstvo púčikov na vetve) sú zoradené do prioritného zoznamu priradeného ku každej osi(obr. 11 b v strede). Jednotlivé púčiky na osi sú považované za *single-metamer* vetvy. *Apicalny* rast je simulovaný premiestnením *koncového púčika* na prvé miesto v zozname, bez ohľadu na množstvo svetla ktoré zachytil. Nakoniec je množstvo svetla rozdelené medzi vetvy a púčiky pomocou váh, v závislosti na pozícii v prioritnom zozname (obr. 11 b v pravo):

$$v_i = v\frac{Q_i w_i}{\sum_{j=1}^{N} Q_j w_j}, i = 1, 2, ..., N \quad (7)$$

kde v je množstvo celkového svetla rozdelovaného medzi púčiky, N je počet púčikov a vetiev pripojených k osi, v_i je množstvo svetla prideleného vetve (alebo púčiku) i a $w_1, .., w_N$ sú váhy. Množstvo svetla priradeného púčiku určuje, koľko *metamersov* vyprodukuje púčik v ďalšom kroku simulácie.

Priradenie veľkej váhy malému počtu produktivných vetiev vytvára model košatejšieho stromu. Širokú rozmanitosť tvarov stromov môžme dosiahnuť odstránením *apikálnho* rastu z kmeňa stromu alebo bočných vetiev. Kontrola *apikálneho* rastu je dôležitá aj pri animácii rastu stromov s hrubým kmeňom.

4.2.7 Pridávanie nových prírastkov

V základnom modeli sú nové výhonky vytvorené v smere púčika. *Koncové púčiky* sú orientované v smere, ktorým je orientovaná vetva, na ktorej sú pripevnené. Orientáciu *bočných púčikov* určuje phyllotaxia a uhol medzi púčikom a jeho materskou vetvou.

Orientácia metamersov tvoriacich nový *prírastok* je ovplivnená dvoma faktormi:

1. optimálnym rastom vzhľadom k ich životnému prostrediu (vektor $\vec{V}$)
2. vplivom gravitácie (tropismus)

Aktuálna orientácia nového *metameru* je tak vypočítana z veľkostí troch vektorov: predvolená orientácia, optimálny smer rastu (ζ) a vektor tropismu (η). Rôzne kombinácie sily vektorov produkujú veľké množstvo rozmanitých foriem stromov (od nízkych hrubých kríkov po vysoké štíhle stromy).

Výhody

- Realistické modely stromov.
- Ovládanie tvaru stromu pomocou váh vektorov.
- Ovládanie tvaru stromu pomocou prerozdeľovania svetla.
- Malý počet vstupných parametrov.
- Možnosť modelovania širokej škály stromov a kríkov.
- Metóda založená na skutočnom biologickom raste stromu.

Nevýhody

- Zložitý model a jeho softvérová implementácia.
- Pri veľkých štruktúrach veľa zložitých výpočtov.
- Zložitá simulácia exotických stromov ako palmy.

Obrázek 12: Ukážka stromov vygenerovaných *Self-organizing metódou* [4]

4.3 Modelovanie podľa predlohy z obrázku

Predošlé metódy modelovania poskytujú realistické možnosti úprav, avšak pre efektívne použitie vyžadujú odborné znalosti ovládania danej metódy. Obvykle dobre fungujú za predpokladu , že primerány tvar vetiev a usporiadania listov sú predvídateľné. Ak však chceme modelovať rôzne neočákavané štruktúry ako napríklad zakrpatenie časti stromu, rôzne degeneratívne ochorenia vetiev a podobne, museli by sme vynaložiť značné úsilie (poprípade pozmeniť celý algoritmus) na dosiahnutie požadovaného výsledku.

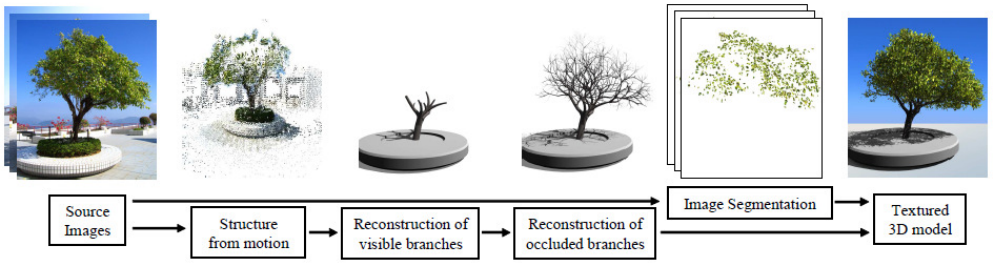
Modelovanie podľa predlohy z obrázku je flexibilnejšie pre modelovanie širokej škály stromov. Môžu sa ňou modelovať všetky typy stromov, aj tie netradičné a degenerované. Na ovládanie tejto metódy nie je potrebné študovať zložité pravidlá a vypĺňať zložité vstupné parametre. Stači len vložiť sériu obrázkov stromu, ktorý chceme vymodelovať.

4.3.1 Metóda Image-based tree modeling

Tento modelovací systém sa skladá z troch hlavných častí:

- Snímanie obrazu a vytvorenie záchytných 3D bodov.
- Vytvorenie vetiev.

- Vytvorenie populácie listov.



Obrázek 13: Kroky metódy *Image-based tree modeling* [1]

Je navrhnutá tak, aby znížila množstvo interakcii užívateľa pomocou vyťaženia maximálneho možného množstva dát zo snímok. Vytvorenie viditeľných vetiev je väčšinou automatické a užívateľ má možnosť vylepšiť ich tvary . Následné vytvorenie listov a konárov, ktoré nie sú viditeľné na obrázku je automatické iba s niekoľkými parametrami zadanými užívateľom. Na generovanie skrytých vetiev stromu sa využiva technika se založená na sebapodobnosti. Parametre vstupujúce do tohto algoritmu sa však nevkladajú uživateľom ale sú extrahované z viditeľných častí stromu na obrázku. Ku generovaniu listov koruny sa ako vsupný parameter zadáva priemer listu. Následňe systém analyzuje segmenty obrázkov podľa farby a pozícia lista v modeli sa vypočíta buď podľa najbližšieho listového segmentu alebo konárového segmentu. Orientácia každého listu je aproximovaná z tvaru oblasti vzhľadom k listu alebo na najvhodnejšie miesto pre list v okolí.

4.3.2 Vstupné snímky a vytvorenie 3D bodov

Ako vstupné parametre sa používa séria snímkov jedného objektu z rôznych uhlov. Objekt by mal byť nasnímaný z čo najrôznejších uhlov od 0 do 360. Táto metóda si vystačí aj s 10 až 20 snímkami každého objektu. Avšak čím viac snímok je k dispozícii, tým jednoduchšie bude vytvorenie 3D bodov.

Pri extrahovaní 3D bodov zo snímok, musí byť skúmaný objekt umiestnený v homogénnom prostredí, aby sa predišlo chybnému vyhodnoteniu bodu pozadia ako bodu objektu. Mračno bodov sa extrahuje pomocou farby pixelov a segmentácie obrázku.

Na samotné extrahovanie a prevod do 3D scény sa používa štandardná metóda opísana v [Lhuillier a Quan 2005]. Pri extrahovaní sa neprenesie celý objekt do 3D scény, pretože je veľká časť objektu zakrytá lístami, konármi alebo je daným algoritmom nerozpoznateľná.

4.3.3 Vytvorenie viditeľných vetiev

Všetky viditeľné vetve sa vytvárajú z prenesených 3D bodov. Vytvoria sa takzvané klastre, kde každá vetva predstavuje jeden klaster. Viditeľné vetve sú rekonštruované pomocou

metódy bottom-up, teda z dola na hor. Model sa začína vytvorením grafu a podgrafov, reprezentujúcych jednotlivé vetvy. Užívateľ vyberie klikom na 3D bod daný klaster, ktorý sa má vymodelovať. To opakuje pokiaľ sa všetky vetvy nevymodelujú. Začína sa od kmeňa smerom k vedlajším vetvám.

Graf G vytvárame tak, že každy 3D bod berieme ako uzol, ku ktorému pripájame jeho susedné body. Susedný bod je ten, ktorý neprekročí prahovú vzdialenosť zadanú užívateľom. Váha medzi rohmi spojených bodov sa vypočíta pomocou vzťahu $d(p,q) = (1-\alpha)d_{3D}+\alpha d_{2D}$ kde predvolene $\alpha = 0.5$. 3D vzdialenosť d_{3D} je euklidovská vzdialenosť medzi p a q normalizovaná ich rozptylom. Pre každý snímok I_i na ktorom sú premietnuté p a q, nech l_i je výsledná priamka v snímku spájajúca ich projekcie $P_i(p)$ a $P_i(q)$. Nech n_i je počet pixelov v l_i a $\{x_{ij}|j = 1, ..., n_i\}$ je množina 2D bodov v l_i. Potom môžme definovať 2D funkcia vzdialenosti nasledovne $d_{2D} = \sum_i \frac{1}{n_i} \sum_j |\Delta I_i(x_{ij})|$, ktorá je normalizovaná pomocou ich rozptylu. V prípade, že v zdrojovom snímku bola vetva identifikovaná a presegmentovaná (použitím nejakej segmentačnej techniky), funkcia vráti nekonečno ak sa nejaká úsečka premieta mimo oblasť vetvy. Každý novo pripojený komponent, sa považuje za vetvový klaster.

Pri konverzii grafu na vetvy, začíname s najnižšie položeným 3D bodom, ktorý bol určený ako súčasť primárnej vetvy (kmeňa). Najkratšie vzdialenosti od koreňových bodov k ostatným bodom sú počítané štandardnými algoritmamy na výpočet najkratšej cesty. Okraje subgrafu sa uchovávajú ak sús súčasťou najkratšej cesty. V opačnom prípade sa vymažú a takisto sa vymažu aj vnútorné body. Tento krok vedie k vytvoreniu prepojenia 3D bodov na povrchu vetvy. Následne sa vypočíta ťažisko bodov v každom segmente a je uvedený ako uzol kostry. Polomer tohto uzla (alebo zodpovednej vetvy) je smerodajnou odchylkou bodov.

Tento model umožnuje uživateľovi taktiež vykonávať niektoré operácie ako napríklad pridavanie alebo mazanie uzlov, vloženie uzla medzi dva susedné uzly, nastavenie polomeru zla a podobne.

4.3.4 Vytvorenie neviditeľných vetiev

Vztvorenie viditeľných vetiev slúži na dva účely : Samotné vytvorenie časti modelu stromu a vytvorené časti sú použité na vytvorenie neviditeľných vetiev. Pri vytváraní vychádzame z predpokladu, že štruktúra vetiev je sebepodobná. V tomto modeli, je každá časť viditeľných vetiev potenciálnym kandidátom replikačného bloku. Hranice koruny sa určia zo vstupných snímkov. V závislosti na dostupnosti zrekonštruovaných 3D bodov, može byť rast "neobmedzený"alebo "obmedzený".

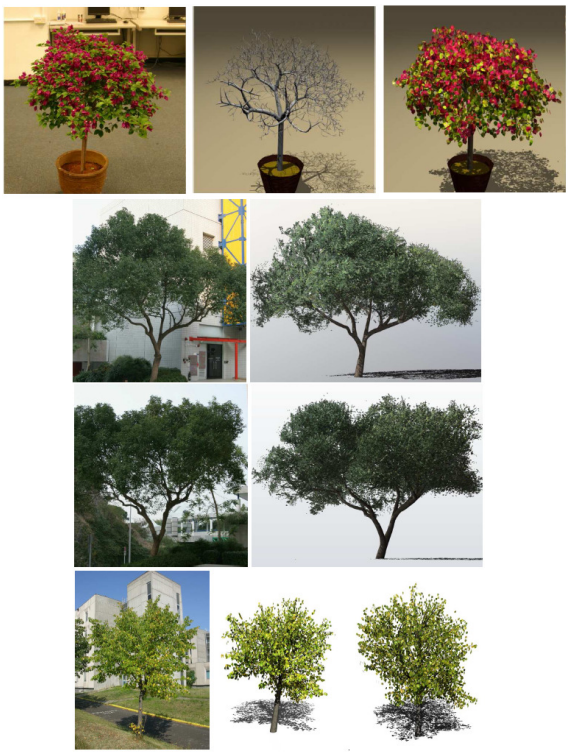
4.3.5 Neobmedzený rast

V oblastiach, kde 3D body nie sú k dispozícii, systém náhodne vyberie koncový bod vetvy alebo uzola a pripojí koncový bod náhodné vybratého replikčného bloku (už vytvorenej vetvy). Hoci výber replikačnej vetvy je väčšinou náhodný, priorita je kladená na silnejšie vetvy alebo tie bližšie ku kmeňu stromu. Pred pripojením replikačného bloku sa vykonávajú 2 operácie: rotácia a zmena mierky. Najskôr je replikačná časť otočená

okolo osi primárnej vetvy o náhodný uhol. Následne je vykonaná zmena veľkosti všetkých častí replikačného bloku tak, aby hrúbka konca replikačnej vetvy bola rovnaká ako hrúbka konca primárnej vetvy. Tento rast je limitovaný siluetou zo zdrojových snímkov a snaží sa ju napodobiť ako najviac sa dá.

4.3.6 Obmedzený rast

Tento rast vetiev sa vypočíta minimalizáciou funkcie $\sum D(p_i, Tree)$ nad všetkými 3D bodmi $\{p_i | i = 1, ..., n_{3D}\}$ kde n_{3D} je počet 3D bodov. $D(p_i, Tree)$ je najmenšia vzdialenosť medzi bodom p a koncovými bodmi vetvy v strome $(Tree)$. Bohužial množstvo všetkých možných podstromov s pevne daným počtom generácii, ktoré majú byť pridané, je exponenciálne. Preto sa používa zavedenie tzv. kužel vplyvu. Aplikuje sa na každý uzol stromu s osou pozdĺž aktuálnej vetvy a s uhlovým rozsahom +- 90° na obe strany. Z tohto uzla sa vyberú iba tie body, ktoré spadajú do kužela pôsobenia. Náš problém sa teda redukuje na minimalizáciu $\sum_{p_i \in Cone} D(p_i, Subtree)$ pre každý podstrom s kužeľom obsahujúcim množinu bodov spojené s podstromom. Ak je kužel prázdny, konáre sú vytvorené pomocou metódy neobmedzeného rastu. Výpočet podstromu sa vykonáva generácia po generácie. Počet generácii pridaných vetiev môže byť kontrolovaný.



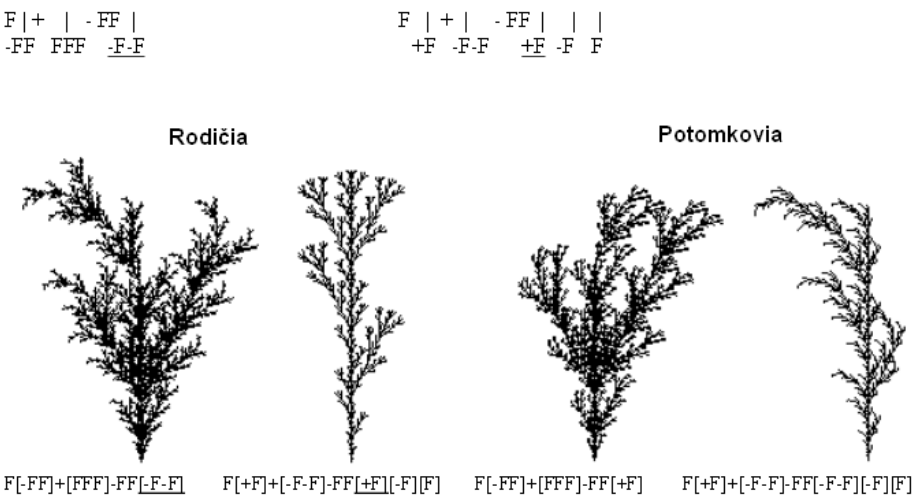
Obrázek 14: Ukážky modelov vygenerovaných pomocou *Image-based tree modeling*[1]

4.4 Genetické algoritmy

Skupina D0L-systémov sa používa na kódovanie virtuálnych organizmov. Chromozóm je reprezentovaný jediným prepisujúcim pravidlom ktorého axiom je vždy F. Pokiaľ chceme používať genetické kódovanie organizmov, musíme najskôr definovať reprodukčné pravidlá. Tieto operácie bývajú väčšinou prepracovanejšie ako tie v klasických genetických algoritmoch. Náš chromozóm má štruktúru odvodenú z gramatiky klasického L-systému. Pre správne odvodenie a interpretovanie genotypu, musia naše operácie produkovať potomkov so správnou syntaktickou štruktúrou. Základ tvoria 3 hlavné operácie: kríženie a 2 typy mutácie.

4.4.1 Kríženie

Navrhnuté kríženie je inšpirované genetyckým programovaním popísaným v (Koza, 1992). Hierarchická reprezentácia rodičov a potmkov, môže vyzerať napríklad takto:



Obrázek 15: Rodičia a potomkovia po krížení. Podčiarknuté časti sú zmenené. [5]

4.4.2 Mutácia

Operácia mutácie predstavuje náhodnú zmenu v štruktúre populácie. V tomto modeli rozlišujeme 2 typy mutácie. Každá z nich pôsobí na odlišnú časť chromozómu.

Mutácia symbolu Náhodná zmena jedného symbolu z množiny $\{F, +, -\}$ v chromozóme za náhodný ale syntakticky správny reťazec.

Mutácia bloku Náhodne vybraný blok symbolov z chromozómu je nahradený náhodným synakticky správnym reťazcom.

4.4.3 Genetický algoritmus

Náš implementovaný algoritmus je v niektorých smeroch odlišný od klasického GA. Za prvé, genotypy majú premenlivú dĺžku a majú pevne definovanú syntaktickú štruktúru. Ďalej je zvolený pevný stav výberu generácie: vždy je vybraná iba 1/5 populácie pre premiestnenie v každej generácie. Nakoniec sú definované vyššie popísané genetické operácie pre každý reprodukčný cyklus.

4.4.4 Fitness funkcia

Niekoľko modelov evolúcie morfologických aspektov umelých organizmov sa zameriava na zložité automatické funkcie meriace čo najvyššiu estetickú alebo funkčnú vlastnosť objektu. Avšak prírodný výber sa neuskutočnuje priamo nad samotnými génmy, ale skôr nad ich efektami na telá organizmov alebo fenotypmi. Práve tu využívame veľký stupeň automatizácie, preto je nevyhnutné vytvoriť adekvátnu fitness funkciu. V našom prípade použijeme sformulovanú hypotézu týkajúcu sa faktorov, ktoré majú najväčší vplyv na evolúciu rastlín. Hypotéza použitá v tomto príklade bola sformulovaná Karlom Niklasom v jeho práci (Niklas, 1985).

Hypotéza sa skladá z nasledujúcich častí:

a) fototropizmus (rast a otáčanie rastlín za svetlom)

b) bilaterálna symetria

c) schopnosť zberu svetla

d) štrukturálna stabilita

e) proporcia bodov vetiev

Bol vyvynitý jednoduchý algoritmus pre quantifikáciu konkurenčných východ sprostredkovaných týmito vzťahmi.

Výber sa odohráva nad fenotypmi. Takže predtým ako môžme ohodnotiť organizmus, musí byť odvodený a geometricky interpretovaný. Každá časť fitness funkcie je quantifikovaná procedúrou, ktorá používa ako vstup geometrické informácie vyprodukované počas kreslenia modelu a vracia reálne čislo medzi 0 a 1. Popis quantifikačnej procedúry:

Uvažujme 2D karteziánsky súradnicový systém s počiatkom v začiatočnom bode modelu. Každý vertex v modeli je reprezentovaný párom súradníc (x,y). Päť vyššie zmienených častí quantifikácie:

4.4.5 Fototropizmus

Štruktúram, ktoré majú vysokú hodnotu pozície Y je priradená vysoká hodnota fitness funkcie. Naopak štruktúram s nízkou pozíciou Y je priradená nízka hodnota fitness funkcie. To predstavuje rast štruktúry smerom hore za svetlom.

4.4.6 Bilaterálna symetria

Balancovaná šírka štruktúry je vypočítaná približne. Absolútne hodnoty vrcholov súradnic x naľavo a napravo od vertikálnej osi sú zvyšované. Inak povedané, čím vybalancovanejšia štruktúra je, tým je výsledok bližšie k 1.

4.4.7 Schopnosť zberu svetla

Schopnosť zhromažďovať svetlo je vypočítaná z veľkosti povrchu listov rastliny (koncových segmentov). Listy zhromažďujú svetlo len ak nie sú zatienené inými listamy. Tieň sa počíta pomocou vertikálnych lúčov smerujúcich zhora nadol.

4.4.8 Štrukturálna stabilita

Vetvy začínajú vždy v štartovnom bode (uzle). Z jedného uzla môže vyrastať aj viac vetiev. Predpokladá sa, že ak štruktúra obsahuje veľa uzlov, z ktorých vyrastá veľa vetiev, je štruktúra nestabilná (procedúra vracia výsledok blízky k 0). Ak naopak štruktúra obsahuje väčšinu "stabilných"uzlov, vracia procedúra číslo blízke 1.

4.4.9 Proporcia bodov vetiev

Celkový počet uzlov, z ktorých vyrastá viac než 1 vetva sa spočíta. Toto číslo je v priamom pomere k počtu všetkých vetiev stromu. Predpokladá sa, že rastliny s vysokým počtom vetiev sú lepšie v zbere svetla a roznášaní semena.

Váha parametrov $(w_a, w_b, w_c, w_d, w_e)$ sa použije pre ladenie efektu všetkých častí procedúry vo finálnej fitness funkcii:

$$F(Phenotype) = \frac{aw_a + bw_b + cw_c + dw_d + ew_e}{w_a + w_b + w_c + w_d + w_e} \tag{8}$$

Pomocou váh $w_a, ..., w_e$ môže užívateľ ovplivňovať celkový vzhľad výsledného modelu.

Obrázek 16: Fitness funkcia s váhami a=100, b=90, c=40, d=20, e=30 [5]

5 Program na generovanie rastlín a stromov

Hlavnou časťou tejto práce je návrh a implementácia programu pre generovanie rastlín a stromov v 3D prostredí a ich následný export vo formáte vhodnom pre iné 3D programy (blender, 3D studio max a podobne). Samotný program je vytvorený v programovacom jazyku C Sharp v prostredí Microsoft Visual Studio 2010 s použitím technológie OpenGL 4. Staršie grafické karty nepodporujú nové OpenGL, takže pri spustení programu na PC so starou grafickou kartou, nemusí fungovať program korektne. Na prepojenie Microsoft Visual Studia s OpenGL 4 som využil knižnice OpenGL4NET.dll, csgl.dll a csgl.native.dll. Program funguje na platforme Windows a bol vyskúšaný na verziách Windows XP, Windows 7 a Windows 8.

5.1 Použité technológie

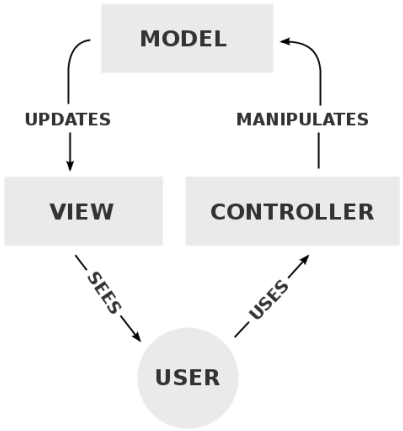
Pre prostredie Microsoft Visual C Sharp som sa rozhodol z niekoľkých dôvodov. Vývoj v jazyku C Sharp je rychlejší ako v klasickom C++, hlavne kvôli tomu, že sa nemusím starať o rušenie dynamický vytvorených objektov. Taktiež odpadla nutnosť tvorby hlavičkových súborov a ich prepájanie medzi jednotlivými triedami a modulmi. C Sharp ponúka tiež najnovšie frameworky a knižnice bez ich pracnej implementácie do projektu. Najväčšou nevýhodou použitia C Sharpu namiesto C++ je rýchlosť programu a prístup do pamäti počas programovania. Program vytvorený v C++ je rýchlejší ako program vytvorený v C Sharp. Taktiež počas programovania v jazyku C Sharp nemáme priamy prístup k smerníkom (musíme využiť blok kódu v *unsafe* móde). Takže hlavným dôvodom využitia C Sharp namiesto C++ je rýchlejší a pohodlnejší vývoj programu na úkor výslednej rýchlosti programu.

Grafické prostredie OpenGL 4 som si zvolil hlavne kvôli jeho rýchlosti, množstvu možností a využívaniu najnovších technológii. Moderné OpenGL (v mojom prípade verzia 4), ponúka vývojárovi množstvo moderných techník programovania v 3D ako sú shadery, práca s novým OpenGL contextom, nové mapovanie textúr pomocou MipMap a mnoho ďalších. Vo svojom projekte využívam aj shadery, konkrétne vertex shader a fragment shader. Používam ich hlavne kvôli zrýchleniu vykreslovania objektov v scéne, kedy sa väčšina výpočtu ohľadom vykreslovania presunie na grafickú kartu. Pre implementáciu OpenGL do Microsoft Visual C Sharp využívam knižnicu OpenGL4NET.dll, čo je knižnica špeciálne vytvorené pre NET technológie. Umožnuje mi prístup a prácu s novým OpenGL a pomocou nej vykonávam takmer všetky OpenGL príkazy. Taktiež používam knižnice csgl.dll a csgl.native.dll, pomocou ktorých využívam funkcie *GLUT*, čo je nadstavba nad OpenGL. Z nej však využívam iba niektoré funkcie na generovanie geometrických objektov a funkciu na prepočet súradnic z obrazovky na súradnice v 3D scéne. Celé jadro programu využíva knižnicu OpenGL4NET.dll.

5.2 Popis štruktúry programu

Pri tvorbe programu som využil architektonický vzor MVC (Model View Controler). Ide o softwérovú architektúru, kedy je celý projekt rozdelený na dátový model, vykreslovacie

rozhranie a riadiacu logiku do troch nezávyslých komponentov. Hlavnou výhodou tohto modelu je, že zmena v jednom komponente, má len minimálny vpliv na ostatné dve komponenty.



Obrázek 17: Model-View-Controller architektúra [7]

V mojom prípade som vytvoril 3 *namespace* moduly: CONTROLER, MODEL, VIEW a jednu triedu clsCreator, ktorá sa stará a o vytvorenie jednotlivých tried v moduloch a ich korektné previazanie.

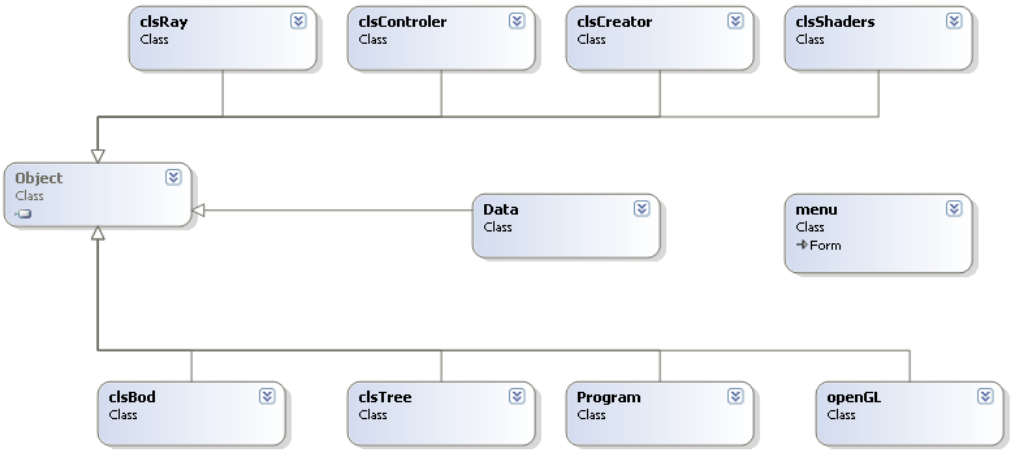
Modul **MODEL** je dátový modul, v ktorom sa uschováva väčšina využívaných dát. Obsahuje triedy clsBod a clsData. Trieda clsBod reprezentuje jeden bod v scéne (či už AP alebo BP) a obsahuje všetky potrebné informácie o jednom bode. Trieda clsData slúži ako dátová trieda a obsahuje skoro všetky využívané dáta. Obsahuje taktiež všetky využívané štruktúry a operácie medzi niektorými. Je v nej uložená napríklad definícia vlastnej štruktúry vektor a operácie s nimi definované alebo štruktúra quaternion a operácie s nimi definované. Všetky údaje sú tu uložené privátne a pristupuje sa k ním pomocou ich properties. Tento modul obsahujú obidva ďalšie moduly a je základným kameňom celého projektu.

Modul **VIEW** je zobrazovací modul. Obsahuje triedy clsOpenGL a clsShaders. Trieda clsShader slúži na vytvorenie, načítanie a skompilovanie shaderov. Taktiež prepojí shadery s programom. Trieda clsOpenGL slúži predovšetkým na vykreslovanie 3D scény. Je hlavným využívateľom knižnice OpenGL4NET a teda aj OpenGL. Snažil som sa všetky úkony súvisiace s OpenGL vložiť do tejto triedy. Obsahuje hlavnú funkciu glDraw, ktorá sa stará o vykreslenie 3D scény. Taktiež obsahuje mnoho vedľajších funkcií súvisiacich s vykresľovaním v OpenGL. Tento modul obsahuje moduly DATA a CONTROLER.

Modul **CONTROLER** je riadiacim modulom. Obsahuje triedy clsControler, clsTree, clsRay, forumlár a hlavnú triedu programu. Hlavnou úlohou tohoto modulu je generovanie stromovej štruktúry a generovanie samotného modelu stromu. Následne uloží vygenerované štruktúry a modely do modulu DATA, odkiaľ si ich prevezme modul VIEW a vykreslí ich v 3D scéne. Tento modul sa taktiež stará o pohyb v scéne a vypočítava sa

tu celý engine programu. Zaradil som tu taktiež uživateľské rozhranie (formulár) a sú v ňom uložené aj eventy 3D scény (MouseDown, KeyUp, Scroll a podobne) Tento modul obsahuje moduly DATA a VIEW. Model VIEW obsahuje predovšetkým kvôli volaniu prekreslenia pri udalostiach na formulári.

Diagram tied programu vyzerá nasledovne:



Obrázek 18: Diagram tried projektu

5.3 Metódy generovania

Program generuje stromy v troch krokoch. Prvým je tvorba koruny a nastavenie parametrov pre generovanie. Druhým je tvorba a prípadna úprava stromovej štruktúry. Tretím je vygenerovanie výsledného stromu a jeho prípadna úprava. Ak ste už v poslednom kroku, je možné sa vrátiť naspäť a výslednú stromovú štruktúru upraviť prípadne doplniť.

5.3.1 Tvorba koruny

V prvom krorku je nutné vytvoriť tvar koruny. V našom prípade sa tvarom koruny myslí rozmiestnenie atrakčných bodov do požadovaného tvaru. Zhruba takýto tvar bude mať koruna výsledného stromu. Atrakčné body reprezentujú množstvo svetla v danom bode. Každý atrakčný bod má určitý dosah. Oblasť, v ktorej sa pôsobenie viacerých atrakčných bodov prekrýva, obsahuje svetlo zo všetkých pôsobiacich bodov. Atrakčné body môžeme vkladať ručne po jednom, alebo hromadne pomocou vytvorenia a zarotovania polygónu, vo vnútri ktorého sa vygenerujú atrakčné body. Generovanie môže prebiehať podľa mriežky, určenej polomerom atrakčných bodov, alebo náhodne. V prípade náhodného generovania je dôležitým parametrom *random koeficient*. Ten určuje, po koľkatích neúspešných pokusoch vygenerovať atrakčný bod, má algoritmus skončiť. Neúspešným

pokusom sa rozumie vygenerovanie atrakčného bodu na koliznom mieste s iným už vygenerovaným atrakčným bodom. Po (prípadne pred) vygenerovaní koruny si môžeme nastaviť parametre atrakčných bodov ako sú priemer, dosah a priemer zničenia.

5.3.2 Tvorba stromovej štruktúry

Pred samotným vygenerovaním stromovej štruktúry je nutné nastaviť parametre bodov stromu. Tieto parametre budú mať podstatný vpliv na výsledný tvar stromu. Taktiež si môžeme zvoliť metódu, ktorou chceme strom generovať. Môj program pracuje s dvoma metódami: *Space colonization* a *Self organization*. Princíp oboch metód je popísaný v texte vyššie v kapitolách 4.2.1 a 4.2.4. Po vygenerovaní stromovej štruktúry už nie je možné vybranú metódu zmeniť. Daný strom bude generovaný vždy výhradne vybranou metódou. Nie je možné kombinovať metódy.

Po úspešnom nastavení parametrov a zvolení metódy, môžeme vygenerovať stromovú štruktúru pomocou tlačidla *Build structure*. Táto štruktúra sa skladá z bodov, ktoré môžeme označovať, posúvať a meniť ich vygenerovaný smer rastu. V tejto fáze si môžeme strom drobne upraviť podľa seba. Môžeme sa aj vrátiť do fázy tvorby koruny a prípadne doplniť do priestoru atrakčné body. Po následnom znovuvygenerovaní štruktúry, by mala štruktúra "dorásť"do priestoru s atrakčnými bodmi (pokiaľ sú koncové listy v dosahu atrakčných bodov). Ak je stromová štruktúra už vygenerovaná, môže obsahovať aj tzv. mŕtve púčiky. Ide o listové body, ktoré z rôznych príčin už nepokračujú v raste. Po znovustlačení tlačidla *Build structure* sa tieto body prebudia k životu a ak majú vhodné podmienky, znovu začnú rásť. Takto môže teda strom pomocou pregenerovania znovu začať rásť aj bez pridania dodatočných atrakčných bodov.

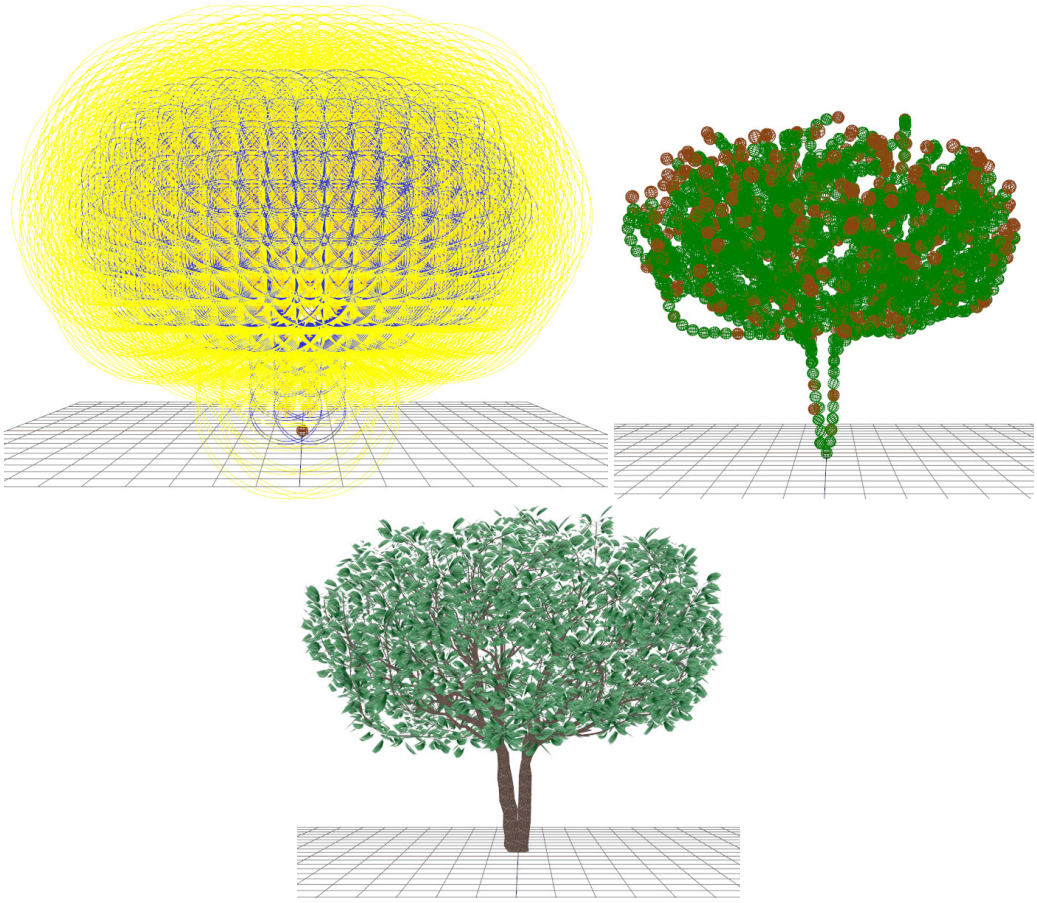
5.3.3 Tvorba modelu stromu

V poslednom kroku budeme generovať výsledný model stromu. Pred samotným generovaním si môžeme nastaviť parametre stromu ako sú zaoblenie, počiatočná hrúbka konárov, veľkosť listov a podobne. Po nastavení parametrov môžeme strom vygenerovať tlačidlom *Build tree*. Vygeneruje sa nám otexturovaný model stromu s listami na konci konárov. Všetky parametre v súčastnej fáze sú interaktívne, čiže ich zmena by sa mala prejaviť ihneď na modele stromu a nemusíte ho teda znovu generovať. V prípade potreby sa môžeme vrátiť aj do predchádzajúcich krokov a previesť úpravy na vylepšenie stromu. Takýmito úpravamy môžu byť napríklad zväčšenie koruny stromov pridaním atrakčných bodov, zmena orientácie rastu konárov (pri prekrútení alebo posune konárov), vymazanie celej vetvy stromu alebo vygenerovanie susedného stromu alebo kríku. Postup tvorby stromu je zobrazený na obrázku 19.

5.3.4 Ďalšia funkčnosť programu

Tento program umožnuje aj exportovať vygenerovaný strom do formátu .obj. Pokiaľ je zvolená voľba pre zobrazenie listov, vygenerujú sa aj listy spolu so stromom. Po naimportovaní stromu do externého programu (napríklad blender) sú listy jeden objekt a kmeň

stromu druhý objekt. Taktiež sa dajú vyexportovať aj vygenerované atrakčné body spolu s ich nastavenými parametrami. Tento export je využiteľný pri porovnávaní algoritmov, kedy oba algoritmy musia mať rovnaké počiatočné parametre (vygenerované atrakčné body). Taktiež sa dá zapnuť funkcia *reporting*, kedy sa po vygenerovaní stromovej štruktúry vytvorí textový súbor (report.txt), kde sú uvedené počiatočné parametre pre generovanie a sledované výstupné údaje (čas algoritmu v milisekundách, počet vygenerovaných uzlov stromu a počet vygenerovaných bodov stromu).



Obrázek 19: Postup tvorby stromu

6 Experimenty

Nasleduje kapitola, v ktorej sa budem zaoberať porovnávaním dvoch implementovaných a popísaných metód generovania stromov. Ide o metódu *Space colonization* a *Self organization*. Obe metódy sú založené na princípe konkurenčného boja o zdroje.

6.1 Meranie metódy Space colonization

Metóda space colonization je založená čisto na kolonizácii voľného miesta. Ide o metódu konkurenčného boja o priestor. Počiatočné semienko rastie v smere najväčšieho pôsobenia atrakčných bodov. Následne vygenerované listové body rastú taktiež v smere najväčšieho pôsobenia atrakčných bodov. Táto metóda nemá v sebe žiaden náhodný prvok, takže pri použití rovnakých vstupných parametrov dostaneme vždy rovnaký výsledok. Vo vytvorenom programe funguje metóda *Space colonization* nasledovne.

Pre každý listový bod sa vypočítaju dva smery rastu. Pre každý smer sa vypočíta aj jeho sila. Následne sa vypočíta rozdiel síl oboch vektorov. Ak je daný rozdiel menší ako hodnota parametru *Divide of leaf point*, nastane rozdvojenie konára. Z toho vyplýva, že parameter *Divide of leaf point* má veľký vplyv na výsledný tvar stromu. Z toho dôvodu som zvolil ako sledovaný parameter pri experimentoch s metódou Space colonization práve *Divide of leaf point*. Následne po rozdvojení konára sa každému novovytvorenému konáru priradí jeden zo smerov. Problém nastal v prípadoch, keď bol jeden smer výrazne silnejší a po rozdvojení konára sa v ďalšom kroku rastu oba konáre nasmerovali na najsilnejší smer rastu. Tento problém som vyriešil tak, že pri rozdvojení sa jenému konáru priradí jeden smer a druhý smer sa zablokuje a naopak. To zaručuje, že nový konár sa bud držať priradeného smeru až do ďalšieho rozdvojenia alebo do dorazenia na miesto najsilenjšieho pôsobenia.

Vypracoval som meranie, kedy boli vygenerované 4 druhy koruny so 4 rôznymi počtami atrakčných bodov. Následne som menil hodnoty parametrov *Divide of leaf point*, *Radius of AP range* a *Radius of AP destruction*. Po vygenerovaní štruktúry som sledoval počet vyprodukovaných bodov stromu, počet uzlov stromu a čas pracovania samotného algoritmu. Vstupné parametre merania sú teda *Divide of leaf point*, *Radius of AP range* a *Radius of AP destruction* a výstupné údaje sú počet bodov stromu, počet uzlov stromu a čas pracovania algoritmu. Výsledky merania sú vedené v nasledujúcich tabuľkách.

Parametre pri generovaní tabulky 1 :

1. Radius of AP = 0,15
2. Radius of AP range = 3
3. Radius od AP destroy = 1
4. Radius of BP = 0,1
5. Max angle = 45

Tabulka1:

Num.	Divide coeficient	AP count	BP count	Nodes	Time [ms]
1	0,5	1912	837	205	187,5
2	0,5	627	117	25	15,625
3	0,5	2695	2227	482	984,375
4	0,5	4229	1213	285	578,125
5	0,05	1912	69	13	31,25
6	0,05	627	65	14	15,625
7	0,05	2695	226	44	109,375
8	0,05	4229	194	39	187,5
9	5	1912	837	205	187,5
10	5	627	117	25	15,625
11	5	2695	2227	482	984,375
12	5	4229	1213	285	578,125

Parametre pri generovaní tabulky 2 :

1. Radius of AP = 0,15
2. Radius of AP range = 5
3. Radius od AP destroy = 2
4. Radius of BP = 0,1
5. Max angle = 45

Tabulka2:

Num.	Divide coeficient	AP count	BP count	Nodes	Time [ms]
1	0,5	1912	1232	360	515,625
2	0,5	627	518	141	78,125
3	0,5	2695	2971	793	1750
4	0,5	4229	4015	1027	3281,25
5	0,05	1912	324	92	250
6	0,05	627	83	24	15,625
7	0,05	2695	1466	354	1078,125
8	0,05	4229	1868	471	1937,5
9	0,1	1912	1794	481	703,125
10	0,1	627	252	73	62,5
11	0,1	2695	2266	606	1281,25
12	0,1	4229	3890	960	3125

Zo získaných dát vyplýva, že hodnotu koeficientu delenia stačí zadávať v rozasu od 0,5 do zhruba 0,05. Pri hodnote koeficientu 5 boli vygenerované dáta zhodné s dátami pri

koeficiente 0,5. Pri koeficiente 0,05, bol vygenerovaný strom príliž riedky a málo rozvetvený. Vygenerovalo sa iba pár dlhých priamých konárov. Optimálna hodnota parametru pre vyvážený strom je teda 0,5. Taktiež parametre *Radius of AP range* a *Radius of AP destruction* mali velký vplyv na tvar stromu. Pri požití hodnoty 5 pre rozsah pôsobenia a hodnoty 2 pre zničenie bol strom rozvetvenejší a mohutnejší. Preto, ak chceme mohnutnejšie stromy, zvolíme vyäčšiu hodnotu pre parameter rozsahu pôsobenia a menšiu hodnotu pre parameter zničenia atraktoru.

Tento experiment som vykonal aj preto, aby som vedel, aké parametre mám použiť pri porovnávaní metód space colonization a self organization, aby boli vstupné parametre pre obe metódy vyvážené.

6.2 Meranie metódy Self organization

Metóda self organization je taktiež založená na princípe konkurenčného boja o priestor. Využíva sa tu ale aj náhodný prvok a kumulovanie a prerozdelovanie svetla medzi všetky aktívne púčiky. V mojom programe je implementovaná táto metóda nasledovne.

Počiatočný púčik v sebe kumuleje silu svetla akou naň pôsobia všetky trakčné body v dosahu. Následne sa vytvorý nový metamér s koncovým púčikom a bočnými púčikmi. Dĺžka metameru závisí od toho, koľko svetla daný púčik naakumuloval. Celočíselná časť púčika určuje dĺžku metameru. Každý metamer sa skladá z jedného hlavného koncového púčika a niekoľko (prípadne 0) vedlajších bodov stromu. Každý vedlajší bod novovytvoreného metameru sa môže stať základným bodom stromu (konára) alebo vedlajším púčikom. Šanca na to, že sa vedlajší bod stane púčikom je $2/3$, z toho vyplýva, že aj pri rovnakých vstupných parametroch, bude výsledok (skoro) vždy iný.

Pri každom kroku rastu sa prejdú všetky púčiky, na ktoré pôsobí svetlo. Toto množstvo svetla sa prenesie do najbližšieho uzla, kde sa kumuluje svetlo zo všetkých púčikov patriaceho danému uzlu. V tomto uzle sa vytvorí zoznam púčikov, ktoré prispeli svetlom. Tento zoznam sa usporiada nasledovne: púčik, ktorý prispel najväčším množstvom svetla sa zaradí na prvé miesto, púčik, ktorý prispel druhým najväčším množstvom svetla bude druhý a tak ďalej. V prípade, že je zvolená možnosť *Apical grow*, budú na prvé miesto zaradené hlavné koncové pučiky oboch konárov, bez ohľadu na to, akým množstvom svetla prispeli do celkového sumáru v uzle. Takýmto spôsobom sa dosiahne štíhleho tvaru stromu. Keď je všetko svetlo zhromaždené v uzle, rozdelí sa pre každý aktívny púčik pomocou vyššie popísanej rovnice číslo 7 v kapitole 3.2.6. Ak je púčiku priradené množstvo svetla menšie ako 1, púčik umiera a stáva sa z neho bod stromu.

Zo vzorca pre prepočet svetla púčikom je zerjmé, že hlavné vstupné parametre tejto metódy sú hodnoty váh a maximálna dĺžka metameru. Ak by množstvo svetla prekročilo hodnotu parametru *Max metamer length*, svetlo sa zníči na hodnotu maximálnej dĺžky metameru podľa vzorca:

$$MetamerLength = min(SilaSvetla, MaxMetamerLength) \tag{9}$$

Preto v nasledujúcom experimente budem sledovať vplyv zmeny vstupných parametrov váhy a maximálnej dĺžky metameru na výstupné dáta. Parametre koruny stromu

sú totožné s prvým experimentom. V experimente budú požité vygenerované atraktory z prvého experimentu.

Parametre pri generovaní tabulky 3:

1. Radius of AP = 0,15
2. Radius of AP range = 5
3. Radius od AP destroy = 2
4. Radius of BP = 0,1
5. Max angle = 45

Tabulka3:

Č.	Weight 1	Weight 2	Weight 3	Max. length	AP count	BP count	Nodes	Time [ms]
1	1,2	0,8	0,6	4	1912	1427	529	500
2	1,2	0,8	0,6	4	627	602	208	78,125
3	1,2	0,8	0,6	4	2695	4225	1456	4734,375
4	1,2	0,8	0,6	4	4229	5586	2027	9140,625
5	0,6	1,2	0,8	4	1912	1925	681	1046,875
6	0,6	1,2	0,8	4	627	978	344	218,75
7	0,6	1,2	0,8	4	2695	4387	1485	5031,25
8	0,6	1,2	0,8	4	4229	5153	1791	7375
9	0,8	0,6	1,2	4	1912	2686	927	1875
10	0,8	0,6	1,2	4	627	830	295	171,875
11	0,8	0,6	1,2	4	2695	4276	1412	4625
12	0,8	0,6	1,2	4	4229	4714	1705	6734,375
13	0,8	0,6	1,2	3	4229	2944	1063	2968,75
14	0,8	0,6	1,2	2	4229	723	257	1062,5
15	0,8	0,6	1,2	5	4229	5716	1993	9140,625
16	0,8	0,6	1,2	6	4229	5846	2027	10968,75
17	1,2	0,8	0,6	3	4229	1962	707	1828,125
18	1,2	0,8	0,6	2	4229	2859	1078	3468,75
19	1,2	0,8	0,6	5	4229	5614	1966	9734,375
20	1,2	0,8	0,6	6	4229	6145	2119	12296,875

V metóde Self organization je náhodný prvok, ktorý spôsobuje, že pri rovnakých vstupných dátach, môťu byť rozdielné výsledky. Preto som meranie robil viac krát ako pri metóde space colonization. Pri tejto metóde by bolo potrebné pre získanie priemerných výsledkov použiť štatistické metódy a opakovať merania niekoľko stoviek krát. Pre naše účely však postačujú získané dáta.

Z dát vyplýva, že hlavný vplyv na počet bodov stromu a počet uzlov stromu má parameter maximálna dĺžka metamerov a samozrejme počet atrakčných bodov. Parametre

váhy majú vplyv skôr na tvar stromu ako na jeho veľkosť. Preto ak chceme regulovať tvar stromu, meníme parametre váhy. Ak chceme regulovať veľkosť stromu, meníme parameter maximálna dĺžka metameru. Čím menšia je hodnota maximálnej dĺžky metameru, tým menšie konáre budú generované a na nich menej púčikov. Následne pri ďalšom kroku bude vygenerované menšie množstvo konárov. A naopak, čím väčšia je hodnota maximálnej dĺžky metameru, tým dlhšie budú konáre a bude vygenerovaných viac púčikov. Tento parameter má samozrejme svoje logické ohraničenie.

Optimálne vstupné parametre pre porovnanie s metódou space colonizaton som zvolil nasledovné:

1. Weight 1 = 0,15
2. Weight 1 = 5
3. Weight 1 = 2
4. Max metamer length = 4

6.3 Porovnanie metód Space colonization a Self organization

V nasledujúcej podkapitole sa budem venovať porovnávaniu implementovaných metód Self organizationa a Space colonization. Hlavným rozdielom je to, že metóda Self organization má v sebe náhodný prvok a metóda Space colonization nie. Preto je poronávanie tých dvoch metód nie celkom objektívne.

Hlavnými sledovanými výstupnými údajmi budú čas pracovania algoritmu a množstvo vygenerovaných bodov stromu. Vstupné atrakčné body budú pre obe metódy rovnaké. Práve tu sa využije export a import atrakčných bodov, kedy sa môžu vygenerované atrakčné body vyexportovať aj s ich parametrami a následne naimportovať. Výsledky porovnávania budú uvedené v tabuľkovej aj grafickej forme.

Nastavenia atrakčných bodov a bodov stromu:

1. Radius of AP = 0,15
2. Radius of AP range = 5
3. Radius od AP destroy = 2
4. Radius of BP = 0,1
5. Max angle = 45

Nastavenia metódy space colonization:

1. Divide of leaf points = 0,5

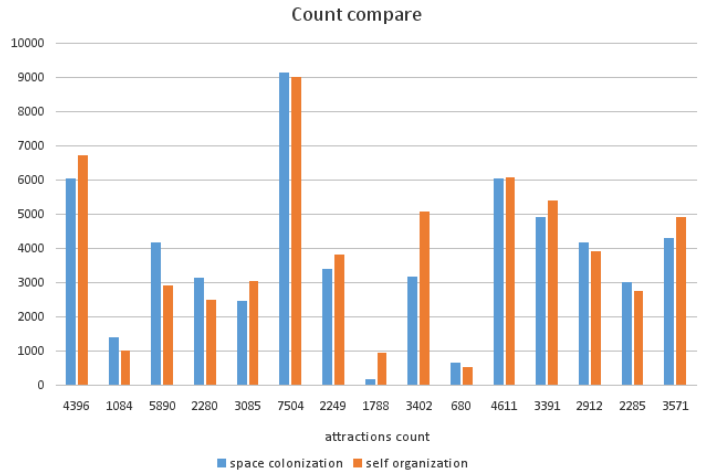
Nastavenia metódy space colonization:

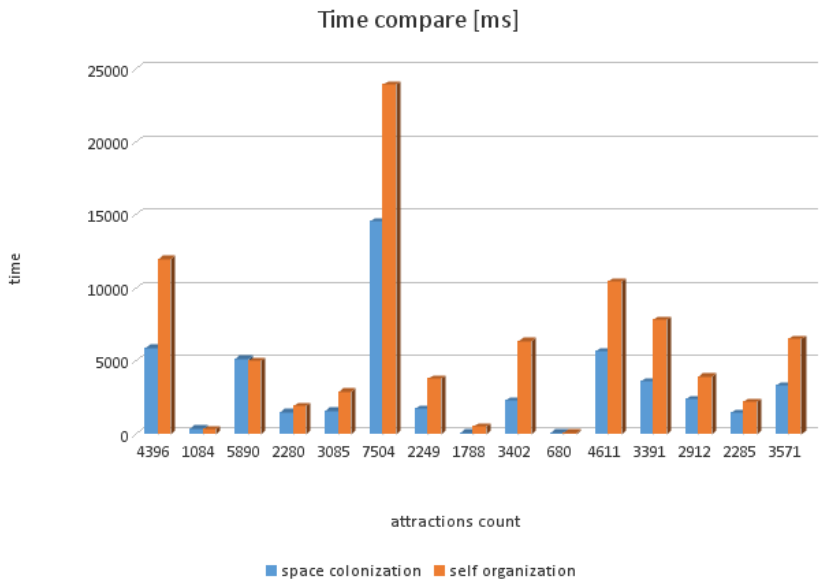
1. Weight 1 = 1,2
2. Weight 2 = 0,8
3. Weight 3 = 0,6
4. Max metamer lenght = 4

Výsledky porovnania metód v tabulkovej forme:

Num.	AP count	BP count-SC	Time[ms]-SC	BP count-SO	Time[ms]-SO
1	4396	6036	5906	6711	11968
2	1084	1395	359	985	343
3	5890	4177	5140	2901	4984
4	2280	3123	1515	2495	1906
5	3085	2451	1578	3041	2890
6	7504	9132	14546	9004	23921
7	2249	3386	1703	3811	3796
8	1788	156	93	946	500
9	3402	3147	2265	5071	6375
10	680	636	93	501	93
11	4611	6029	5640	6080	10453
12	3391	4893	3609	5373	7843
13	2912	4172	2390	3917	3921
14	2285	2986	1468	2755	2203
15	3571	4306	3343	4897	6484

Výsledky porovnania v grafickej forme: Výsledky porovnania v grafickej forme:





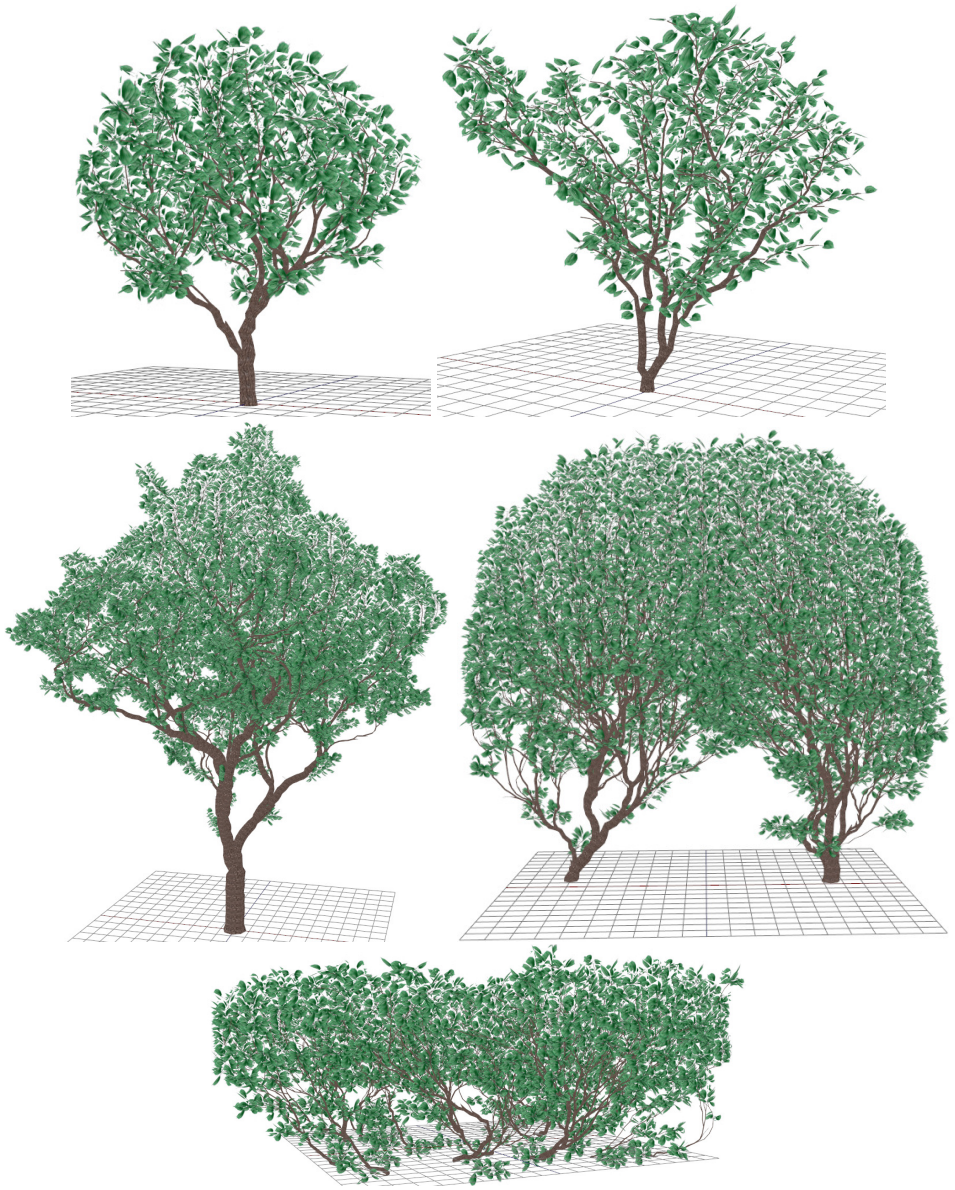
Z výsledkov je zrejmé, že metóda self organization je časovo náročnejšia než metóda space colonization pri približne zhodnom počte vygenerovaných bodov stromu. Avšak metóda self organization nám ponúka širšie možnosti ovplivnenia výsledného tvaru stromu. Pri metóde space colonization môžeme tvar stromu ovplivniť tvarom vygenerovanej koruny (atrakčných bodov) a koeficientom delenia. Metóda self organization nám ponúka možnosť úpravy výsledného tvaru stromu tvarom koruny stromu, nastavením váh a určením maximálnej dĺžky metameru.

Hlavný rozdiel medzi porovnávanými metódami je teda v rýchlosti algoritmu a v možnosti ovplivnenia tvaru výsledného tvaru stromu. Metóda space colonization je rýchlejšia, ale ponúka nám menšie možnosti ovplivnenia výsledného tvaru stromu. Metóda self organization je pomalšia, ale má viac možností ovplivnenia výsledného tvaru stromu.

6.4 Ukážky vygenerovaných stromov mojím programom

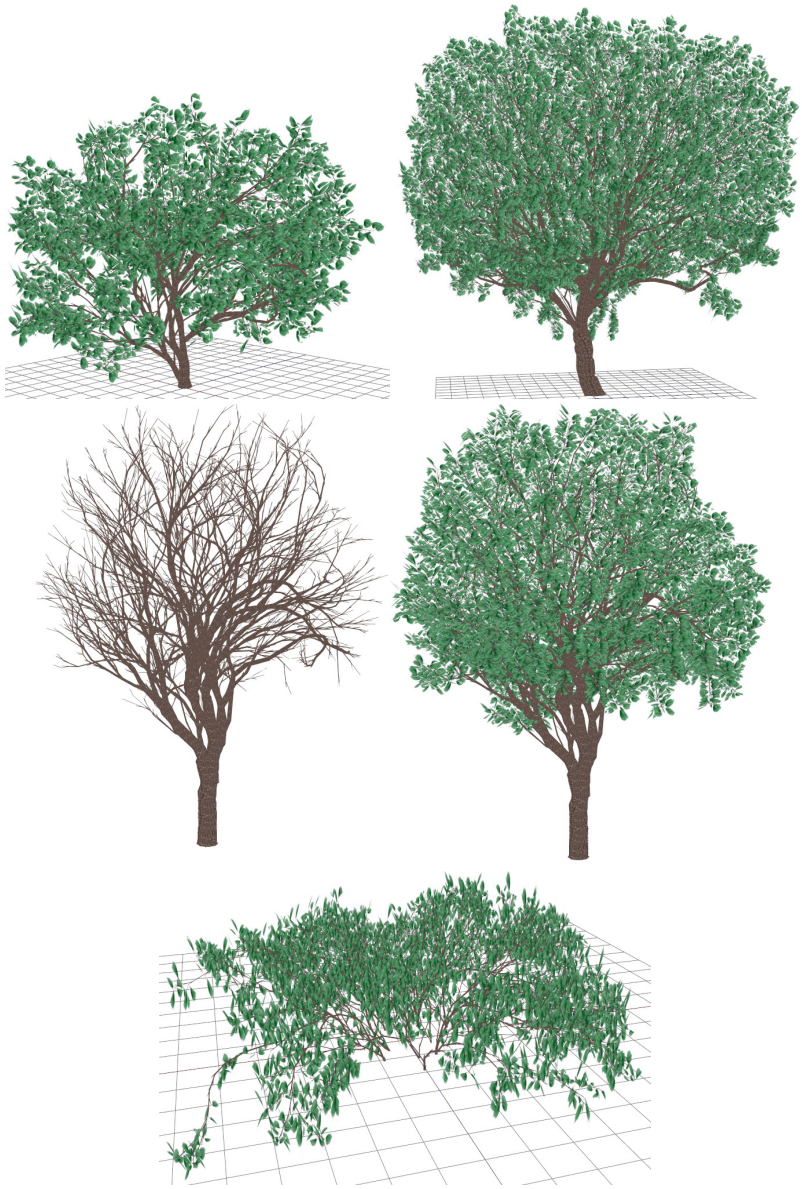
Nasleduje kapitola, kde budú predvedené ukážky vygenerovaných stromov mojím programom s použitím oboch algoritmov. Môžte vyzuálne porovnať vygenerované stromy a taktiež bude predvedené vizuálne porovnanie so skutočnými stromami.

6.4.1 Stromy vygenerované metódou Space colonization



Ako vidno z obrázkou, touto metódou sa dajú ľahko generovať realistické stromy mierneho pásma, Taktiež sa dajú generovať kríky, väčšie stromy, či stromy bojujúce o zdroje svetla. Nevýhodou tejto metódy je, že rast sa simuluje iba ťahaním posledného bodu za svetlom, čo nezodpovedá reálnemu rastu stromov. Napriek tomu, táto metóda generuje veľmi pekné a použitelné stromy.

6.4.2 Stromy vygenerované metódou Self organization



Metóda self organization je založená na raste konárov z púčikov umiestnených na konároch stromu. To má za následok realistickejšie vyzerajúce stromy a kríky. Táto metóda sa viac drží reálneho biologického rastu stromov ako metóda space colonization. Taktiež nám poskytuje možnosť lepšie ovládať výsledný tvar koruny (nastavením váh, popríp ade zapnutím možnosti apikálneho rastu). Všetky tieto vlastnosti robia z metódy self organization vynikajúci nástroj na generovanie reálnych stromov a kríkov.

V príkladoch vygenerovaných stromov, som používal jeden typ textúry pre strom a tiež jeden typ textúry listov. Textúry je možné v programe zmeniť a tým dodať stromom ešte realistickejší vzhľad, poprípade vytvoriť konkrétny druh stromu (breza, buk, lipa). Taktiež je možné stromy vyexportovať (či už s listami alebo bez) do súboru formátu .obj a následne naimportovať do externého grafického programu (blender, 3D studio max, ...) a v ňom upraviť podľa potrieb (zmeniť textúry, materiál, osvetlenie a podobne) a tým ešte zvýšiť realistický vzhľad stromu.

7 Záver

Počas tvorby diplomovej práce som získal mnoho teoretických i praktických skúseností s vývojom v 3D grafike aj s tvorbou modelou stromov. Ako prvé som musel naštudovať problematiku tvorby 3D modelov stromov a následne moje vzdelávanie pokračovalo naštudovaním vývoja v C Sharp, OpenGL 4 a v 3D prostredí ako takom. Vďaka tomu sa moja odborná znalosť v oblasti programovania rozšírila o pre mňa dôležitý jazyk C Sharp a veľmi zaujímavé OpenGL.

Na základe získaných znalostí som vyhotovil program, ktorý je schopný generovať realistické 3D modely stromov v reálnom čase a následne ich vyexportovať do súbora pre použitie v inom (prepracovanejšom) 3D programe. Taktiež boli vykonané experimenty a porovnanie nad dvoma významnými metódami generovania stromov. Z výsledkov experimentov vyplýva, že metóda *Space colonization* je rýchlejšia avšak neponúka toľko možnosti detailnejšie riadiť výsledný tvar stromu ako metóda *Self organization*. Taktiež pri tvorbe modelu celkom nevychádzka z biologického rastu stromov. Vo výsledku však vytvára pekné reálne modely stromov. Metóda *Self organization* vychádza viac z biologického rastu stromov a tým generuje realistickejšie tromy než metóda *Space colonization*. Taktiež nám ponúka viac možností upraviť výsledný tvar stromu. Všetky tieto klady majú však za následok pomalšie generovanie modelov než metóda *Space colonization*. Rozdiel však nie je nijako veľký, rádovo niekoľko sekúnd.

S výsledkom práce som vcelku spokojný, aj keď by sa ešte dalo niečo vylepšiť. V priebehu času by som chcel ešte odstrániť drobné nedostatky a doplniť funkčnosť programu. Napríklad by som chcel vylepšiť editáciu vygenerovanej štruktúry stromu, pridať možnosť pregenerovať strom a vylepšiť systém napájania vetiev. Po úprave popísanej funkčnosti by som rád ponúkol tento program ako voľne šíriteľný nástroj na generovanie 3D modelov stromov. Verím, že nájde uplatnenie u začinajúcich tvorcov hier či filmov, alebo u každého, kto by si chcel rýchlo a jednoducho vygenerovať 3D model stromu, alebo kríku.

8 Reference

[1] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, Long Quan *Image-based Tree Modeling*, The Hong Kong University of Science and Technology

[2] Adam Runions, Brendan Lane, Przemyslaw Prusinkiewicz *Modeling Trees with a Space Colonization Algorithm*, Department of Computer Science, University of Calgary, Canada

[3] Sören Pirk, Ondrej Stava, Julian Kratt, Michel Abdul Massih Said, Boris Neubert, Radomír Mech, Bedrich Benes, Oliver Deussen *Plastic Trees: Interactive Self-Adapting Botanical Tree Models*, University of Konstanz, Germany, Purdue University, USA, Adobe Systems Inc., USA

[4] Wojciech Pałubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomır Mech, Przemyslaw Prusinkiewicz *Self-organizing tree models for image synthesis*, University of Calgary, Adobe Systems Incorporated

[5] Gabriela Ochoa *On Genetic Algorithms and Lindenmayer Systems*, COGS–School of Cognitive and Computing Sciences, The University of Sussex

[6] Wikipedia *L-systém*, http://sk.wikipedia.org/wiki/L-systém [online]

[7] Wikipedia *Model–view–controller*, http://en.wikipedia.org/wiki/Model-view-controller [online]

Zoznam príloh

- CD s kompletným obsahom práce a vytvoreným programom